Novel Feature Selection Algorithms for Improving Neural Network Performance

Zongyuan Zhao M.Sc., B.Sc.

Submitted in fulfilment of the requirements for the degree of Doctorate of Philosophy at the School of Engineering and ICT, University of Tasmania (December, 2016)

Declaration

I, Zongyuan Zhao, do hereby declare that this thesis contains no material that has been accepted for the award of any other degree or diploma in any tertiary institution, except by way of background information and duly acknowledged in the thesis. To the best of my knowledge and belief it contains no material previously published by another person, except where due reference is made in the text of the thesis, nor does the thesis contain any material that infringes copyright.

Signed: .....

Date: 21 / Sep / 2016

Authority of Access

This thesis may be made available for loan and limited copying in accordance with the *Copyright Act 1968*.

Signed: .....

Date: 21 / Sep / 2016

Statement of Co-authorship

The following publications form part of the work that was undertaken for this thesis:

Zongyuan Zhao (Candidate)
Shuxiang Xu (co-author)
Byeong Ho Kang (co-author)
Mir Md Jahangir Kabir (co-author)
Yunling Liu (co-author)
Rainer Wasinger (co-author)

Zongyuan Zhao, Shuxiang Xu, Byeong Ho Kang, Mir Md Jahangir Kabir and Rainer Wasinger are within the School of Engineering and ICT, University of Tasmania.

Yunling Liu is within the College of Information and Electrical Engineering, China Agricultural University.

Zhao, Z., Xu, S., Kang, B. H., Kabir, M. M., Liu, Y., & Wasinger, R. (2016). Utilizing Feature Selection on Higher Order Neural Networks. In M. Zhang (Ed.), *Applied Artificial Higher Order Neural Networks for Control and Recognition* (pp. 375-390). Hershey, PA: Information Science Reference. doi:10.4018/978-1-5225-0063-6.ch015

Zongyuan Zhao (70%) is the primary author. He conducted the research and prepared the material for publication. Dr. Shuxiang Xu (10%), Dr. Byeong Ho Kang (5%), Mr. Mir Md Jahangir Kabir (5%), Rainer Wasinger (5%) of the School of Engineering and ICT, University of Tasmania, and Dr. Yunling Liu (5%) of the College of Information and Electrical Engineering, China Agricultural University, all of them provided general guidance and editing as supervisors and colleague.

Zhao, Z., Xu, S., Kang, B.H., Kabir, M.M.J., Liu, Y. & Wasinger, R., 2015. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. *Expert Systems with Applications*, *42*(7), pp.3508-3516.

Zongyuan Zhao (70%) is the primary author. He conducted the research and prepared the material for publication. Dr. Shuxiang Xu (10%), Dr. Byeong Ho Kang (5%), Mr.

Mir Md Jahangir Kabir (5%), Rainer Wasinger (5%) of the School of Engineering and ICT, University of Tasmania, and Dr. Yunling Liu (5%) of the College of Information and Electrical Engineering, China Agricultural University, all of them provided general guidance and editing as supervisors and colleague.

Zhao, Z., Xu, S., Kang, B.H., Kabir, M.M.J. and Liu, Y., 2015. Instance Selection and Optimization of Neural Networks. *Information Technology In Industry*, *3*(1), pp.1-9.

Mr. Zongyuan Zhao(70%) is the primary author. He conducted the research and prepared the material for publication. Dr. Shuxiang Xu (15%), Dr. Byeong Ho Kang (5%), Mr. Mir Md Jahangir Kabir (5%) of the School of Engineering and ICT, University of Tasmania, and Dr. Yunling Liu (5%) of the College of Information and Electrical Engineering, China Agricultural University, all of them provided general guidance and editing as supervisors and colleague.

Zhao, Z., Xu, S., Kang, B.H., Kabir, M.M.J. & Liu, Y., (2014). Investigation of Multilayer Perceptron and Class Imbalance Problems for Credit Rating. *International Journal of Computer and Information Technology*, 3.4 (2014): 805-812.

Mr. Zongyuan Zhao(70%) is the primary author. He conducted the research and prepared the material for publication. Dr. Shuxiang Xu (15%), Dr. Byeong Ho Kang (5%), Mr. Mir Md Jahangir Kabir (5%) of the School of Engineering and ICT, University of Tasmania, and Dr. Yunling Liu (5%) of the College of Information and Electrical Engineering, China Agricultural University, all of them provided general guidance and editing as supervisors and colleague.

We the undersigned agree with the above stated proportion of work undertaken for each of the above published or submitted manuscripts contributing to this thesis.

Signed:	Signed: .....
.....	Date: 20th September 2016
Dr. Shuxiang Xu	Prof. Andrew Chan
Supervisor	Head of School
School of Engineering and ICT	School of Engineering and ICT
University of Tasmania, Australia.	University of Tasmania, Australia.

Abstract

Data mining and machine learning have become enormously pivotal in this Big Data time, as people are tremendously eager to predict from what they have known, and foresee the unknown. During the process of machine learning, selecting features that are genuinely useful and helpful to the prediction tasks remains a central issue, because a smaller but more relevant dataset can improve the performance of machine learning.

In this thesis, the whole procedure of feature selection is investigated, beginning with a new data preparing method, Average Random Choosing Method (ARCM), which can solve Class Imbalance Problem. Experimental results show that ARCM can significantly improve the prediction accuracy of Artificial Neural Network (ANN) for imbalanced datasets, in comparison with other researches' results.

After that, the new filter named Consistency Concentration Based Feature Selection (CCBFS) is introduced. It is based on the conception of consistency based feature selection (CBFS). CCBFS can evaluate each individual feature and works with both nominal and continuous features, which used to be a shortage of consistency based algorithms.

A GA based wrapper is proposed after CCBFS. This Accumulate Elitism Genetic Algorithm (AEGA) searching approach for feature selection inherits the advantages of GA based methods: quick parallel searching and large searching space. AEGA also covers the limitation of GA such as slow converging speed. Especially for ANN, AEGA has good resistance on the unstable performance of MLP and shortens the training times.

In the end, the hybrid structure Multi Combined Filter-Wrapper Feature Selection (MCFWFS) is introduced. This novel algorithm uses CCBFS to make a pre-selection and accelerate converging process for AEGA. The multi combined structure ensures fast computing time and large improvement of predictor's performances at the same time. It is designed for a wide range of real world problems, even with extremely large datasets.

The main contributions of this research include a novel data preparing method (ARCM), a novel filter for feature ranking (CCBFS), a novel GA based wrapper (AEGA) and a novel structure of hybrid feature selection (MCFWFS). Experiments have demonstrated that these new algorithms have outstanding performances on many kinds of datasets and real world problems in the Machine Learning categories of classification and multi class recognition. This research provides new solutions for improving the machine learning performances and make up shortages of current feature selection methods at some extent.

Acknowledgements

I would like to express my sincere gratitude to those who gave me the continuous support and cooperation during my PhD study.

I would like to thank my supervisor Dr. Shuxiang Xu, for his guidance, encouragement and effort throughout this project. Without him, I would never be enrolled in PhD study or finish it. Thanks must also go to my second supervisor, Assoc Prof. Byeong Ho Kang, for his many suggestions and useful discussions, and Dr. Leonie Ellis for her valuable advices of this thesis. Thanks to Dr. Yunling Liu for her kind suggestions during the initial stages of this research. Also thanks to the University of Tasmania and the School of Engineering and ICT, for providing me the scholarship and facilities.

Many thanks to Mir Md Jahangir Kabir, Dr. Rainer Wasinger, Dr. Mark Brown, Dr. Simon Stannus and Amanda Lunt, for those useful discussions and helps. Also thanks to all the staffs in school for helping me go through these years.

Finally, thanks to my family and friends for your encouragements and trusts. Thanks to my wife who can always give me warm sunshine in the darkest nights.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Introduction

Artificial Neural Network (ANN) is one of the most popular data mining algorithms with a long research history (Taktak and Lisboa, 2006). Inspired by human brain and neurons, theory about ANN was proposed by Warren McCulloch and Walter Pitts as a computational model based on mathematics in 1943 (McCulloch and Pitts, 1943). As computers were not sophisticated enough to handle ANN model in early days, theories about it cannot solve any real-world problem until back propagation (BP) learning algorithm appeared in (Werbos, 1975). Nowadays, as the unprecedented increment of data, ANN has shown its outstanding ability in processing highly nonlinear classification problems (Hong-Gui et al., 2013). Applications in real world that based on ANNs include both classification and clustering problems, such as regression, pattern recognizing, system control, financial application, text detection and many other applications (Sporea and Gruning, 2013). Recently, ANN has shown its specially ability in Deep Learning (Hinton et al., 2012).

As the time of Big Data is coming, data explosion makes datasets highly dimensional and with large amount of instances. However, as to solving specific problems such as classification, clustering, regression and recognizing, irrelevant and redundant features aggravate computing complexity (Dayhoff, 1996). These large datasets require higher performance of ANN, including lower computing time and better generalisation ability. To solve this problem, this research aims at improving the performance of ANN, which meets the essential requirement of ANN applications.

From the background research about ANN, the following approaches are most commonly considered for improving ANN speed and generalisation abilities (Dayhoff, 1996):

- Instance Selection
- Feature Selection
- Architecture
- Learning Algorithms

Among these, Feature Selection, Architecture, and Learning Algorithms can improve both speed and generalisation abilities, while Instance Selection is needed for handling

imbalanced datasets, and therefore, Instance Selection is usually considered a preliminary research for the other three approaches (Garc et al., 2014).

Class imbalance problem exists in many real world datasets where one class of data comprises considerably more samples than others. As ANN models assume that unseen data (test data) has the same distribution as the training data, the class imbalance problem can significantly decrease classification accuracy (Minlong et al., 2013). Pre-processing data by using Instance Selection can solve this problem effectively (Garci et al., 2013).

Feature selection is a key issue in reducing computational time and improving generalisation abilities in solving machine learning problems in the general categories of classification, clustering, pattern recognition, regression, and others (Baccianella et al., 2014, Zenglin et al., 2010, Ramona et al., 2012). It is especially significant in the current Big-Data era when the generalisation abilities of computational models are usually compromised due to over-fitting (Yamada et al., 2014).

Architecture complexity of ANN is highly related to computing complexity, as simple architecture with less neuron and weights in ANN need less computing time. Optimize of architecture may include model pruning algorithm, minimum number of hidden nodes and modifying active function (Silvestre and Lee Luan, 2006). Another important issue of improving ANN's generalization ability is the learning algorithm. Approaches that can improve learning algorithms exist in optimize initial connection weights and convergence rate (Sheng and Banta, 2006). Assessment measure is also vital to learning algorithm on the aspect of choosing the best one.

This research will focus on developing new feature selection methods, with exploring Instance Selection for imbalanced datasets as the preliminary research. Time permitting, architecture and learning algorithms will also be researched to further improve ANN performances.

1.2 Background

Feature selection, also known as attribute selection, or variable selection, is the process of selecting a subset of relevant features for use in computational model construction (Chakraborty and Pal, 2014, Kwak and Chong-Ho, 2002). The rationale with using a

Feature selection technique is that a lot of datasets contain many redundant or irrelevant features. Redundant features are those which provide no more information than the currently selected features, and irrelevant features provide no useful information for establishing models. Benefits provided by feature selection techniques include improved model interpretability, shorter training times, and enhanced generalisation by reducing over-fitting (Mingkui et al., 2013, Luping et al., 2010).

Typically a feature selection method includes four basic steps: subset generation, subset evaluation, stopping criterion and result validation (Dash and Liu, 1997a). The data flowchart is described in Figure 1

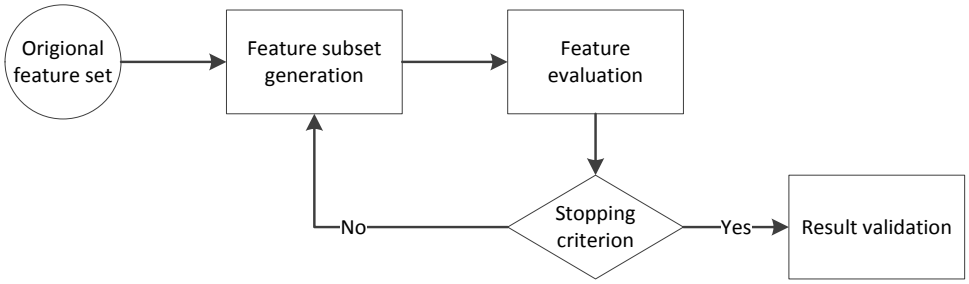

Figure 1 Data flowchart of feature selection

Traditional feature selection was achieved by experts in special fields who have empirical knowledge. As data accumulated in a speed unmatchable by human's capacity of data processing, automatic feature selection is essential to data mining since it can have great affection on reducing computation time and improving mining performance (Guyon and Elisseeff, 2003). Basically, feature selection methods can eliminate irrelevant features or find out important features, which decrease the amount of features in some extent.

Feature selection is utilized in many data mining areas such as classification, clustering, association rules and regression. Applications include text categorization, image retrieval, financial prediction, genomic analysis and so on (Ting et al., 2010). Actually, it can be treated as a search procedure: search for an acceptable feature subset evaluated by certain criterion from the original dataset. Thus, utilizing an effective searching strategy is essential for feature selection methods.

As information explosion brings excessive amount of data, data processing becomes essential to successful data mining. In some datasets, there could be more than 1000

features that represent different characteristics of a single model. For instance, in the Leukemia data set, there are 7071 attributes. These high dimensional datasets seriously aggravate computational complexity of both data mining models and feature selection approaches.

In addition to the time consuming problem brought by Big Data, class imbalance can also hinder the performance of both ANN and feature selection algorithms. In classification and pattern recognition problems, some classes may only have less than 10% of all instances. Assessments using these datasets may be biased and have negative effect on feature selection algorithms' performances. Thus, class imbalance problem should be solved before further feature selection research is conducted.

This research aims at improving the performance of ANN by feature and instance selection approaches. However, there are several shortages in the current feature and instance selection algorithms. First, instance selection methods that can solve class imbalance problems delete or add instances, which may remove useful information or bring in noises. In feature selection research, as those wrappers for ANN models need too much computing time, there is very few valid feature selection methods that can improve ANN performance. This research will develop novel algorithms to cover these shortages.

1.3 Motivation

From the brief introduction of ANN and feature selection above, it is clear that enhancing the efficiency and performance of ANN is with great value, even a tiny improvement can benefit our lives regarding to so many practical usages. For instance, in credit card application assessments, a credit scoring model is used to judge if the applier is qualified or not. If an algorithm can raise the scoring accuracy 1% then the loss of banks will decrease and also the complaints from customers. Efficiency is important in this case as well, for the reason that there may be thousands of appliers inputting their personal information into the application system, which will be a heavy burden for the banks' IT services. If some irrelevant or redundant personal information can be removed without any affection on credit prediction, lots of credit application work can be saved.

As machine learning can solve so many practical problems, there are numberless examples that can prove the importance of enhancing the performance, or to be more accurate, the prediction ability of ANN. Research has shown that different learning approaches of machine learning have similar results, and have indifferent impact on the performance of the prediction models (Langley and Simon, 1995). Data, on the other side, have enormous impact on the performance. All predictions made by machine learning are based on the hypothesis that new data have similar statistical regularity as the data used in learning (Hall, 1999). So the pre process of training data is with high research value and this thesis is only focus on this issue.

Data, consisted of instances and attributes, can be refined by experts according to their experiences. However, in many circumstances the data are too complex for experts or even no expert in some new cases. Thus, the instance selection and feature selection can only utilize the statistic regularity in data.

The first problem of dataset that hinders performance of machine learning is the Class Imbalance problem. The class imbalance problem is a challenge to machine learning and data mining, and it has attracted significant research associated with data mining (Wasikowski and Chen, 2010). This problem can appear in many real world datasets, where one class comprises considerably more samples than the other one or more classes. A classifier affected by the class imbalance problem for a specific data set would see strong accuracy overall but very poor performance on the minority class. Experiments demonstrated that class imbalances hinder the performance of ANN classifiers (Japkowicz, 2000). It is also observed that as the complexity of the domain increases, ANNs are more sensitive to imbalance problem. Thus, in this research, class imbalance problem should be solved before discussing any other issues, and Instance Selection should be taken as a preliminary research.

Existing feature selection methods for machine learning typically fall into two broad categories: filters and wrappers. Filters evaluate the worth of features using the learning algorithm that is to ultimately be applied to the data (Hall, 1999). Advantages of filter include faster computation and stable to various kinds of machine learning model. Current filters presented in literatures contain some drawbacks. Some filters can only handle with numeric or nominal features, which make them impossible to be used on

many practical datasets. Also, many filters only focus on the consistency of each feature, but ignore the concentration that also determined the quality of features. Although some consistency based filters have shown their ability on feature subset selection, evaluation for each features and feature ranking list is also important. For these reasons, a new filter is designed in this thesis to overcome these shortages.

Wrappers, on the other side, can eliminate redundant features and improve a specified machine learning approach at some extent. Disadvantages of most wrappers exist in time consuming, as the learning procedures should be done many times in wrappers. Also the current GA searching approach is slow when converging to local optimum. Several optimization of GA involves more parameters, which evokes the already difficult problem of GA: parameter setting. To solve these problems, a new wrapper is designed in this thesis.

In the end, a hybrid feature selection method that includes both filters and wrappers is designed, and utilized after solving Class Imbalance problem of datasets. This hybrid method can be used to improve performances of ANNs in the aspect of refining training data.

1.4 Research Problems and Questions

From the literature review, there are limitations of the existing feature selection methods for ANN. To improve the performance of ANN by feature selection, there are several problems to be solved.

1. Class imbalance problems in datasets can greatly affect the performance of ANN.
2. There is no filter specially designed for hybrid feature selection.
3. The existing feature subset searching strategies in wrappers are not suitable to ANN.
4. Hybrid feature selection models cannot achieve fast speed and high performance at the same time.

According to these research problems, 4 questions need to be answered in this research.

1. How to solve the class imbalance problem by a preliminary data preparing process?
2. What is required for the filter used for hybrid feature selection? And how to design a novel filter that can both with high performance and suitable for hybrid feature selection?
3. As ANN training takes long time, how to design a fast searching process for wrapper?
4. How to combine filters with wrappers to achieve an effective hybrid feature selection method?

1.5 Research Aims and Main Contributions

The overarching aim of this research is to improve the performance of ANN by feature selection. This aim can only be achieved by answering all research questions in the last section. To be more specific, the aim of each step in this research is listed as follows.

1. Design a novel instance selection method for data preparation. This method should neither generate more instances nor abandon existing instances, but should have efficiency on solving class imbalance problem for binary classification.
2. Design a filter for hybrid feature selection. This filter should be able to select out features that are relevant to the target class, so it can be used as a pre-selection for wrappers. Also it should be a feature ranking method and the feature ranking list should be used to accelerate wrappers.
3. Design a fast searching process for ANN wrapper. Finding a feature subset that has higher prediction accuracy is the basic aim. As ANN training always takes long time, this new wrapper cannot calls the fitness function too much times.
4. Design hybrid feature selection method by combining filter and wrapper together. This hybrid feature selection must with high computing speed and good performance regarding to enhance the prediction accuracy for ANN.

The contributions of this research are consistent with the aims. The most important contribution is providing a novel hybrid feature selection method that can improve the

performance of ANN. At the same time, the achievement of each process is also important and can be used in other works.

Main contributions of this research are listed as follows.

1. Provide a novel instance selection method for data preparing. The new method ARCM can enhance the prediction accuracy of ANN without adding or removing instances from the original dataset.
2. Design a novel filter called CCBFS. CCBFS is based on Consistent Feature Selection method. It can evaluate each feature individually and provide a ranking list according to the importance (the consistency concentration rate).
3. Design a novel searching strategy for ANN wrapper. This AEGA searching strategy is based on genetic algorithm. The advantages of AEGA include faster converging speed and less fitness calling. It is especially suitable for ANN.
4. Design a novel hybrid feature selection method MCFWFS. This method is a multiple combination of filters and wrappers. It can improve the performance of many kinds of ANN.

1.6 Structure of This Thesis

This section briefly introduces the thesis in the following way:

Chapter 2 summarizes the essential background information including basic concepts of machine learning, instance selection, feature selection and other relevant issues.

Chapter 3 briefly describes the research methodologies used in this thesis.

Chapter 4 focuses on the new designed Instance Selection method for solving Class Imbalance problem. The design of ARCM will be described in detail.

Then a new filter called CCBFS is demonstrated in Chapter 5. Its basic concept, concentration based feature selection, will be introduced first. Then other sections will show the new ideas and optimized parts in CCBFS comparing with the classical CBFS, along with the motivations and advantages.

After that, a GA based wrapper, AEGA, is shown in Chapter 6. Similar with the last chapter, the basic idea of GA and SGA for feature selection is introduced, as part of them will still be used in AEGA. Then a detailed description of AEGA will be shown.

The hybrid feature selection algorithm, MCFWFS, will be introduced then in the end of this chapter. Two different combination ways of CCBFS and AEGA will be described. Then the overall structure of MCFWFS, including the preprocessing algorithm ARCM, will be demonstrated.

Chapter 7 describes all experiments in this thesis, including ARCM, CCBFS and MCFWFS. Datasets details, evaluation methods and experiment environments will be shown first. Each section ends up with a summary of the corresponding experiment.

The last chapter gives out a discussion and conclusion. Future work is also described in this chapter.

References used in this thesis are listed in Bibliography part.

1.7 Summary

This research develops novel algorithms that can improve performance of ANN on two aspects: instance selection methods to solve data imbalance problem, and feature selection methods to find highly relevant attributes. The new algorithms will cover shortages of current methods such as very low computing efficiency of ANN wrappers. To solve the four research problems that are generated by background researches, four corresponding aims are proposed, and four contributions will be given in the end of this research. The most important aim and contribution is improving the performance of ANN.

In the next chapter, background researches of ANN, instance selection and feature selection will be given.

Chapter 2 - Literature Review

Parts of chapter 2 have been published as: Zhao, Z., Xu, S., Kang, B. H., Kabir, M. M. J., Liu, Y., Wasinger, R., 2016. Chapter 15. Utilizing feature selection on higher order neural networks (in) Zhang, M. (ed.): Applied artificial higher order neural networks for control and recognition, Information Science Reference, Hershey, PA, pp. 375-390. ISBN 9781522500636

The published portions have been removed for copyright reasons.

2.1 Introduction

The last section explains that the main aim of this research is to design algorithms for improving ANN performances. To achieve this target, novel feature selection algorithms will be designed. This chapter will establish the backgrounds and supports of this research through reviews of relevant literatures.

It starts from the basic concepts and classical algorithms of ANN, including MLP, HONN and several classic algorithms. Current researches indicate that ANNs can achieve high prediction accuracy for some real world problems. MLP is a typical ANN learning algorithm but long training time is one of the shortages. HONN is an improvement of MLP and proved to be effective in recent years, for those dataset with higher order correlations. However, all these kinds of ANNs' performances still have large improving spaces. Higher prediction accuracies and shorter learning time are the aim of this research.

To improve the performance of ANN, the class imbalance problem is solved by instance selection methods in this research. Both resampling and embedded methods are not fit for this research, because all of the current methods add or delete instances. This may have side effect on the later feature selection process, since it may remove important feature relation information, or add into irrelevant information.

Feature selection is seen as a very effective way to improve machine learning performances. Consistency based filter methods have short computing time and obvious improvement for machine learning, but it can provide no feature ranking results which is vital in this research. GA based wrappers are proved to be more effective on the given machine learning algorithms comparing with filters, but for ANN feature selection it is still too computational complex. To design a novel hybrid feature selection algorithm for ANN, both filters and wrappers need to be modified.

2.2 Machine Learning Algorithms

Machine Learning plays an important role in data mining and artificial intelligence. Every machine learning algorithm has its advantages and limitations. In this research, several ANN models, such as MLP, HONN and RBF will be used as the learning machine to assess the performance of feature selection. These three typical ANN

algorithms can handle real world problems with different type of datasets and scales. For filters, the classification accuracy of these models is used to evaluate the performance of each filter algorithm. In wrappers and hybrid feature selection methods, these ANN algorithms are also used in the feature selection process to guide the search.

Other machine learning methods, such as decision tree and Naive Bayes classifier, are used in the test of filters. Performances on different learning machines are important evaluations to judge the effectiveness of filters.

2.2.1 Artificial Neural network (ANN)

The research of ANN has been started in the late 1940s, when psychologist Donald Hebb invented an unsupervised learning named Hebbian learning (Hebb, 2005). Then the Hebbian learning was realized by computational machines in 1950s. Although Frank Rosenblatt proposed the idea of "perceptron" in late 1950s (Rosenblatt, 1958), the learning process of it was not known until after Paul Werbos created backpropagation (BP) algorithm (Werbos, 1974). BP overcomes the shortages of early ANNs, such as incapable of solving exclusive-or problem and long running time, and is still used as a very popular training algorithm nowadays (Yuchun, 1991). ANN received even more attention since the 21st century, with the advent of deep learning research and big data problems (Hinton et al., 2006).

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use (Haykin, 1999). It was motivated by inspecting the human brain, which has high efficiency in computing and recognizing(West, 2000). According to the structure of networks, it can be divided as single layer feed forward networks, multilayer feed forward networks and recurrent networks. Feed forward networks do not have any feedback loop, which are different from recurrent networks. This thesis only focuses on feed forward networks.

ANNs are made of neurons, or called simple perceptron. The structure of neuron is shown in Figure 2. Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realise nonlinear function, called

neuron activation function. Signal *e* is adder output signal, and y = f(e) is output signal of nonlinear element. Signal y is also output signal of neuron.

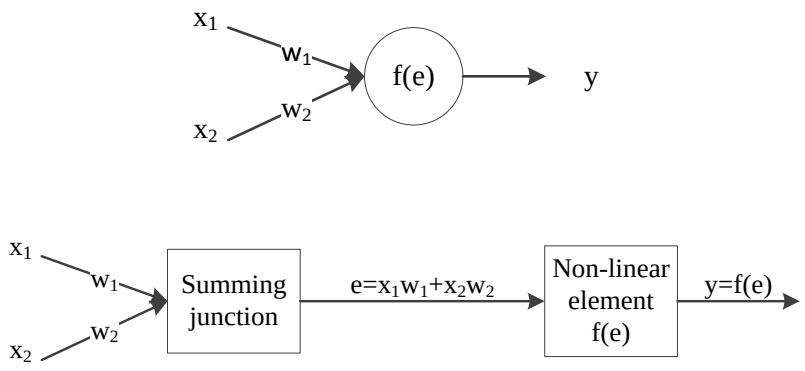

Figure 2 Structure of neuron

A typical feed forward network is multilayer perceptron (MLP). Other kinds of feed forward networks include single layer networks, radial basis function networks, support vector machine and others. Among them, MLP has better performance when applying on some models, especially for complex models and nonlinear classification problems (Karkkainen and Heikkola, 2004).

To better solve nonlinear problems with ANN, the Higher Order Neural Network (HONN) was created (Schmidt and Davis, 1993). It has good performances on those datasets with nonlinear relationships.

2.2.2 Multilayer Perceptron (MLP)

A typical three-layer perceptron includes input layer, hidden layer and output layer. A typical three layer MLP is shown in Figure 3. Input layer accepts input instances directly without any modification. The second layer has many neurons and the number of it may have influence on the performance of whole network. Each hidden neuron adds inputs firstly and then realise activation function. Thus it gets an output value (or signal) which is also the input of the next layer. Output neuron works similar with hidden neurons, but the output of it represents the output of the whole network.

Figure 3 removed

Figure 3 A typical three-layer MLP model

Neural networks cannot work without “learning”. The learning procedure is realized by learning algorithms, which calculate the weights of each neuron in the network. The most popular learning algorithms for feed forward networks include back propagation (BP) (Yuchun, 1991). The algorithm of BP utilized on a three-layer network is described as follows:

```
Initialize network weights (often small random values)
Do
        for each training example ex
                prediction = neural-net-output(network, ex)  // forward pass
                actual = teacher-output(ex)
                compute error (prediction - actual) at the output units
                compute Δw_h for all weights from hidden layer to output layer  // backward pass
                compute Δw_i for all weights from input layer to hidden layer   // backward pass
continued
                update network weights
until all examples classified correctly or another stopping criterion satisfied
return the network
```

Usually there is no activation function for neurons in the input layer and the output neurons are summing units. Then the output of a three layer MLP is

Portion removed

The typical learning algorithm of MLP takes very long running time, as all training instances need to be calculated many times. According to some basic tests of this research, for a dataset with 20 to 30 features and more than 500 instances, the training process takes about 30 seconds on PC, without using 10 fold validations and any data pre-processing methods. For those datasets with more features and instances, this training process will be much longer.

To solve this problem, several improvements have been created, and Radial Basis Function Network (RBF) is one of the most used.

2.2.3 Radial Basis Function Network (RBF)

Radial Basis Function Network (RBF) is a kind of ANN using radial basis functions as the activation functions (Haykin, 1998). It was first designed in paper (Broomhead and Lowe, 1988) and has been applied to many fields, such as clustering (He and Liu, 2009, Loong et al., 2008) and classification (Marcos et al., 2007, Cinar and Sahin, 2010). Similar with a three layer MLP. It also has three layers: input layer, hidden layer and output layer. As MLP uses BP to train the network, RBF calculate the weights of neurons only once in training (Chandrashekar and Sahin, 2014). The output of RBF is calculated by

$$\mathrm{Y} = \sum_{i=1}^{M} w_i \rho(\left|\left|X - c_i\right|\right|)$$

(2,2)

where $j = 1$ to M, and M is the number of neurons in hidden layer. c_i is the centre vector for neuron i, and w_i is the weight of connection between hidden neuron i and the output neuron. $\rho(||X - c_i||)$ is usually a radially symmetric Gaussian function given by

$$\rho(||X - c_i||) = e^{\frac{||X-c_i||^2}{\sigma^2}}$$

(2.3)

Weights of hidden neurons w_i will be calculated by training algorithms. Its vector W can be calculated by

$$W = \rho^{-1} \cdot \mathrm{Y}$$

(2.4)

where matrix ρ is consisted of the radial basis function value ρ_j for the ith instance.

2.2.4 Higher Order Neural Network (HONN)

Portion removed

2.2.5 C4.5 Decision Tree (J48)

Decision Tree algorithm is used to test the performance of filters in this thesis, as filters should be able to enhance the prediction performance of different learning machines. It uses a decision tree as a predictive model. In a decision tree structure, leaves represent class labels and branches represent conjunctions of features that lead to those class labels (Liu et al., 2015). There are many algorithms to build a decision tree, and C4.5 is one of the typical algorithms invented by (Quinlan, 1993). C4.5 is often referred to as a statistical classifier, as it shows good performance on solving classification problems (Sravani et al., 2014).

C4.5 builds decision tree with the concept of information entropy. The algorithm of C4.5 is as follows (Kotsiantis et al., 2007):

Check for base cases
For each attribute *a*
    Find the normalized information gain ratio from splitting on *a*
Let *a_best* be the attribute with the highest normalized information gain
Create a decision node that splits on *a_best*
Recur on the sub lists obtained by splitting on *a_best*, and add those nodes as children of node

An implementation of C4.5 can be found in Weka with the name of J48.

2.2.6 Naïve Bayes Classifier

Naive Bayes classifier is also used to test the performance of filters along with C4.5 decision trees. It applies Bayes' theorem with strong assumption of independence between features (Mahalakshmi and Sivasankar, 2015). It is based on the concept of conditional probability:

$$\mathrm{p}(C_k|x) = \frac{\mathrm{p}(C_k)\mathrm{p}(x|C_k)}{\mathrm{p}(x)} \tag{2.7}$$

For two class labels *i* and *j*, under condition (inputs) X, the label with higher conditional probability is more likely to be the actual label. This can be achieved by calculating

$$R = \frac{P(i|X)}{P(j|X)} = \frac{P(i)P(X|i)}{P(j)P(X|j)} \tag{2.8}$$

If R>1 then predict *i*, else predict *j*.

An implementation of Naive Bayes can be found in Weka.

2.2.7 Summary

In this section, the structure and learning algorithm of ANN are introduced. Some typical ANN structures are shown along with Naive Bayes and Decision Tree and will be used in the experiment of filters to demonstrate filters' abilities. From this background research it is easy to see that training data has huge impact on the training of ANN. Thus, both instance selection and feature selection can improve the performance of ANN by enhancing its training process.

As a classic type of ANN, MLP is used in this research for solving a wide scope of real world problems. RBF and HONN are also utilized for some special kinds of problems. Naive Bayes and Decision Tree as two classic machine learning algorithms are used for comparison in filter experiments.

2.3 Instance Selection

Instance Selection is the process to generate input instance group from the original datasets. This process can be used for generating training-validation-test instances groups and avoid class imbalance problems (Liu, 2010). Instance Selection approaches that can solve class imbalance problems are divided into two categories: resampling and embedded methods.

2.3.1 Resampling

Resampling techniques aim at correcting problems with the distribution of a data set. Weiss and Provost (Weiss and Provost, 2003) noted that the original distribution of samples is sometimes not the optimal distribution to use for a given classifier, and different sampling techniques can modify the distribution to one that is closer to the optimal distribution.

Resampling methods include over sampling and under sampling. Under-sampling of the majority (normal) class has been proposed as a good means of increasing the sensitivity of a classifier to the minority class. However, as most under sampling methods do not consider the relationship between examples, data redundancy or information loss may easily occur (He and Garcia, 2009). Thus, over-sampling methods are more popular in recent years.

One widely used resampling method, Synthetic Minority Over-sampling Technique (SMOTE) is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement(Chawla et al., 2011). It can achieve better classifier performance (in ROC space) than only under-sampling the majority class.

However, SMOTE generated the same number of synthetic examples for each minority example and this strategy may cause data overlapping. Some methods, which can overcome this limitation of SMOTE have been proposed, such as borderline-SMOTE (Han et al., 2005) and Adasyn (Haibo et al., 2008). Borderline-SMOTE only oversamples the borderline examples of the minority class. Adasyn adapts the number of synthetic examples for every minority example according to the distributions.

The oversampling methods try to overcome the property of imbalanced class distribution by adding examples to the training set. However, the duplicating or generating of examples may make the training set noisier and cause over-fitting(He and Garcia, 2009). Furthermore, adding training examples will also increase the training time. To overcome these drawbacks, data cleaning techniques were proposed. Tomek link is a useful definition for cleaning data. It can be used to clean up data after an oversampling method, such as the Random Over Sampling, SMOTE, and Adasyn (He and Garcia, 2009).

Some research also reported that resampling methods cannot improve NN performances or even hinder them. Some other experiments demonstrate that complex resample methods such as SMOTE are not better than the simply Random Over Sampling (Bhowan et al., 2013, Khoshgoftaar et al., 2010, Khoshgoftaar et al., 2011).

2.3.2 Embedded Methods

Another category of solutions is the embedded methods. These methods embedded a resampling approach in model training process, or modify the training algorithms and network structures. A typical embedded solution for imbalanced dataset problem is Dynamic Sampling Approach (Minlong et al., 2013). The main idea of it is selecting examples for training in each epoch to avoid redundant information and to make the best use of the training data. Main steps of this algorithm can be described as follows:

(1) Randomly fetch an example x from the training set.
(2) Estimate the probability p that the example should be used for training.
(3) Generate a uniform random real number μ between 0 and 1.
(4) If $\mu < p$, then use x to update the MLP.
(5) Repeat steps 1-4

The DyS merged over sampling methods into the training process of MLP, while fixed the over fitting problems. However, it is just an adaptive form of random over sampling by duplicating both minority and majority samples. The basic idea is the same as resampling methods but the experiment results are better in the aspect of MLP generalization abilities.

Cost-sensitive method is another kind of embedded methods for class imbalance (Castro and Braga, 2013). They always follow two steps:

(1) Set a cost matrix for the class imbalance problem to formulate the problem as a cost-sensitive problem, and
(2) Employ a method to solve the cost-sensitive problem.

Boosting is also widely used to solve imbalance problems (Galar et al., 2012). This kind of data mining model has advantages as it can resample the data space automatically which eliminates the extra learning cost for exploring the optimal class distribution and the representative samples.

2.3.3 Summary

All researches covered in this background research need to add or delete instances to keep a balance between classes. Even some embedded methods prefer to use some

instances rather than others. This may cause information loss or adding into irrelevant information, which is highly likely to damage feature selection process such as filters.

To solve the class imbalance problem without adding or delete instances, a novel instance selection algorithm will be created in this research. As a kind of resampling method, it focuses on distributing instances for training, validation and test randomly.

2.4 Feature Selection

Portion removed

According to evaluation function, wrappers can also fall into two groups: The partial derivative based saliency measure and the weight based saliency measure.

Nowadays, hybrid feature selection methods that contain both wrappers and filters are welcomed by researchers for its higher improvement of models generalization abilities. It can mainly be divided into two groups based on the joint structure. One way is to use filters before wrappers as a pre selection process. Another one is to merge filters into wrappers, which mainly focus on using filters in local search of wrappers to enhance the efficiency. The first one is more suitable for big data problem, while the later one is obviously more eligible on improving models' performances.

2.5 Filters for Feature Selection

Filters are wildly used for their high efficiency and great enhancement of model's accuracy. Filters show their strong ability in eliminating relevant features, especially when the number of introduced features is high and redundant features occur (Kersten,

2014). Except eliminating relevant features, filters can also select the most relevant features when utilized on supervised learning.

Several tests had proved that filters can have huge impact on classification accuracy of MLP models, and MLP model is also more sensitive to filters compared with Naive Bayers and decision trees (Karabulut et al., 2012). As filters take no consideration of data mining models, a filter algorithm can be used for all kinds of models while the performance is unpredictable. The most important character of a filter is its evaluation function, which presents the character of optima feature subsets (Liu and Yu, 2005). Basically it includes distance measures, information measures, dependency measures and consistency measures. In this research, some popular filters will be introduced. Some of them will be used in experiments and compare with the filter proposed in this thesis.

2.5.1 Basic Consistency Measure

The earliest idea of consistency based feature selection was quite simple: find out a subset of features that is consistent with the target feature. FOCUS (Almuallim and Dietterich, 1991) was first introduced by Almuallim and Dietterich in 1991 and then improved it to FOCUS2 in (Almuallim and Dietterich, 1994). This algorithm stops the search in the first set of features that fully consistent with the class label. Although it can guarantee finding the smallest consistent feature subset, there are also some limitations. First, FOCUS can only be applied on discrete features. Discretization algorithms must be applied on continuous features before measuring. Several improvements of FOCUS are proposed, such as CFOCUS and F (Arauzo et al., 2003), to adapt to continuous features. Second, noise data may greatly hinder the performance of FOCUS. The algorithm has to add another feature just because of one inconsistent data, and that may be redundant. Another limitation is the low searching speed, as a random searching strategy cannot guide the search.

Two consistency based feature selection are widely used. One is created by Liu, Motoda and Dash (Liu et al., 1998) as Consistency based Feature Selection. This is a feature subset selection method based on the idea of FOCUS but with more efficient searching strategy. It is a feature subset selection method designed for nominal feature selection.

Another one is Rough Set Feature Selection (RSFS). It is a group of filters based on rough set theory that is also based on the concept of consistency.

2.5.1.1 LVF Consistency Based Feature Selection

In (Liu et al., 1998) Liu, Motoda and Dash proposed a probabilistic approach – Las Vegas Algorithm for feature subset selection (LVF). LVF makes probabilistic choices to help guide them more quickly to a correct solution (Liu et al., 1998). The LVF algorithm is described as follows:

```
Input:  MAX-TRIES
        D: dataset
        N: number of attributes
        γ : allowable inconsistency rate
Output: sets of M features satisfying the inconsistency criterion
C_best = N
for i=1 to MAX-TRIES
        S = randomSet(seed);
        C = numOfFeatures(S);
        if(C<C_best)
                if(InconCheck(S, D)< γ )
                        S_best = S; C_best = C;
                        print_Current_Best(S);
        else if((C=C_best)and(InconCheck(S, D)< γ ))
                print_Current_Best(S);
end
```

The algorithm starts with a random selected feature subset S in each round. It will do nothing with the situation $C>C_{best}$, which means the number of selected attributes must decrease if not equal to the current best subset.

The inconsistent criterion is defined as follows:

(1) Two instances are considered inconsistent if they match except for their class labels;

(2) For all the matching instances (without considering their class labels), the inconsistency count is the number of the instances minus the largest number of instances of class labels

(3) The inconsistency rate is the sum of all the inconsistency counts divided by the total number of instances.

In their later researches, the inconsistency rate is also introduced to simplify the algorithm. The inconsistency rate and consistency rate are defined as follows:

$$Inconsistency\ Rate = \frac{number\ of\ inconsistent\ instances}{number\ of\ instances} \tag{2.9}$$

$$Consistency\ Rate = 1 - Inconsistency\ Rate \tag{2.10}$$

In Liu's algorithm, the inconsistency is compared with a pre-specified rate γ. Unless specified, the default value of γ is 0. If the inconsistency rate is below γ, then it means the subset of features is acceptable.

The goal of this algorithm is to find a feature subset that contains no inconsistent. That feature subset may not the smallest subset, and algorithm stops when it finds the first consistent subset. Hash table is used to accelerate the algorithm. To make it adapts to continuous features, datasets contain continuous features must be discretised before applying consistency based feature selection.

Antonio (Arauzo et al., 2003) pointed out that the computation can be very fast by using hash table. Through a direct search, all instances are grouped according to their value of attributes. Then the number of inconsistency is the number of instances that do not belong to the majority class. In the average case, of this process is in $O(n)$.

2.5.1.2 Rough Set Feature Selection

According to Antonio's paper about consistency measures (Arauzo et al., 2003), Rough Set Feature Selection can also be regarded as one kind of consistency based feature selection.

Rough Sets was first defined by Zdzisław I. Pawlak in this paper (Pawlak, 1982) and later book (Pawlak, 1991). Given a data table $S=(U, A)$ where the universe U is a finite nonempty set of objects and A is the set of attributes. Every attribute $a \in A$ associates a

set V_a, of its values, called the domain of a. A data pattern of S is any feature value vector $v=(v_1,\ldots v_n)$ where $v_i \in v_{ai}$ for i=1,…n such that $v=a(x)$ for some $x \in U$ (Parthalain et al., 2010).

Each nonempty subset B ⊆ A determines an indiscernibility relation, which is defined as: $R_B = \{(\mathrm{x},\mathrm{y}) \in \mathrm{U} \times \mathrm{U} \mid a(x) = \mathrm{a}(\mathrm{y}), \forall \mathrm{a} \in \mathrm{B}\}$. The relation R_B partitions U into some equivalence classes given by $\mathrm{U/R}_B = \{[x]_B | x \in U\}$, or just *U/B*, where $[x]_B$ denotes the equivalence class determined by x with respect to B, i.e., $[x]_B = \{y \in U \mid (x,y) \in R_B\}$ (Jiye et al., 2014).

Given an equivalence relation R on the universe U and X ⊆ U, the lower approximation and upper approximation of X are defined (Wang and Zhou, 2009) by

$$\underline{R}X = \cup \{x \in U | [x]_R \subseteq \mathrm{X}\} \; or \; \underline{R}X = \{x \in U | [x]_B \subseteq \mathrm{X}\} \tag{2.11}$$

and

$$\overline{R}X = \cup \{x \in U | [x]_R \cap \mathrm{X} \neq \emptyset\} \; or \; \overline{R}X = \{x \in U | [x]_B \cap \mathrm{X} \neq \emptyset\} \tag{2.12}$$

The tuple $< \underline{R}X, \overline{R}X >$ composed of the lower approximation and upper approximation is called a rough sets.

A decision table is a data table $\mathrm{S} = (\mathrm{U}, \mathrm{C} \cup \mathrm{D})$ with $\mathrm{C} \cap \mathrm{D} = \emptyset$, where *C* is called a condition set and *D* is called a decision set. Given P ⊆ C and $U/D=\{D_1, D_2,\ldots D_r\}$, the positive, negative, and boundary region of *D* with respect to the condition attribute set P is defined by

$$\mathrm{POS}_P(D) = \cup_{k=1}^{r} \underline{R_P} D_k \tag{2.13}$$

$$\mathrm{NEG}_P(D) = U - \cup_{k=1}^{r} \overline{R_P} D_k \tag{2.14}$$

$$BND_P(D) = \cup_{k=1}^{r} \overline{R_P} D_k - \cup_{k=1}^{r} \underline{R_P} D_k \tag{2.15}$$

To achieve feature selections, the attribute dependency is an important issue that can discover which attributes are strongly related to which other variables. In rough set theory, dependency is defined in the following way:

For $P, Q \subset A$, the attribute dependency $\gamma_P(Q)$ is defined as:

$$\gamma_P(Q) = \frac{|\mathrm{POS}_P(Q)|}{|U|} \leq 1 \tag{2.16}$$

If $\gamma_P(Q) = 1$, that means Q totally depends on P, and if $\gamma_P(Q) = 0$, then Q does not depend on P.

The Rough Set Feature Selection is based on the idea of removing attributes without affecting the dependency degree. That means, the selected feature sets R have the same dependency $\gamma_R(D)$ as the original condition attributes. A quick feature selection algorithm based on Rough Set theory includes two parts: finding the core, and then get the reduction.

A reduct is a minimal subset of attributes that can fully characterize the knowledge in the datasets. Formally, in the data table $S = (U, C \cup D)$, a subset R is a reduct of C if:

(1) $\gamma_R(D) = \gamma_C(D)$ and
(2) $\forall a \in R$, $\gamma_{R-\{a\}}(D) \neq \gamma_R(D)$

It should be mentioned that a reduct may not be the subset with smallest cardinality. Also the reduct of a data table is not unique. The interaction of all reducts is called the core. All attributes in the core cannot be removed without causing a decrease of dependency.

The RSFS algorithm is described as follows.

```
RSFS(C, D)
C is the set of all conditional attributes
D is the set of decision attributes
Set R = ∅
for (i=1;i<|C|;i++)
```

```
   if ( γ_{C−{a_i}}(D) < γ_C(D)) then R = R ∪ {a_i}
while (γ_R(D) < γ_C(D))
   C=C-R
   for (i=1;i<|C|;i++)
      if ( γ_{R∪{a_i}}(D) > γ_R(D)) then R = R ∪ {a_i}
   end for
return R as the reduct
```

In the RSFS algorithm, the core is found firstly by checking the dependency of all attribute sets with only one attribute missing. If the dependency changed, then it means the missing feature is essential to keep the dependency, in other words it is in the core. Then starting from the core, other attributes are added in if they can increase the dependency. The whole algorithm stops if the dependency of R is equal to the original condition set C, which means R can completely represent C. It should be noticed that this method cannot find the "global" minimum reduct, but can save computing time at some extent regarding to generate a proper reduct.

2.5.2 Other Consistency Based Feature Selection

Antonio et.al proposed an Inconsistent example pairs (IEP) measure in their paper (Arauzo et al., 2003). When two instances have different target values but the same values in all other features, then these two instances form an inconsistent pair. They redefine the inconsistent measure by the number of inconsistent pairs. That is:

$$Inconsistency\ Rate = \frac{number\ of\ inconsistent\ pairs}{number\ of\ instance\ pairs} \tag{2.17}$$

$$Consistency\ Rate = 1 - Inconsistency\ Rate \tag{2.18}$$

The number of instance pairs equals to $\frac{n(n-1)}{2}$ where n is the number of instances.

The algorithm of IEP is similar with Liu's, only replacing the consistency measure with IEP measure. The empirical evaluation of IEP, Liu's consistency measure and wrappers

demonstrate that consistency measures can achieve similar results to the wrapper approaches with much better efficiency.

Zhao and Liu (Zhao and Liu, 2009) propose a feature scoring metric based on data consistency, and develop a filter algorithm named INTERACT to select relevant features while implicitly exploring feature interaction. The basic idea of INTERACT is described in the related work section.

In (Boros et al., 2000) a consistency based measure is used in an embedded feature selection. The consistency measure is redefined to handle with numerical features and missing data.

In (Kuncheva, 1992), a Fuzzy Rough Set Feature Selection (FRFS) was proposed which combines fuzzy set with rough set theory. Fuzzy set theory is introduced to deal with the continuous features, with the consideration of taking a discretization step before RSFS.

Recently, more fuzzy based consistency measure is researched. In (Chakraborty and Chakraborty, 2013), the authors proposed a new Fuzzy Consistency Measure (FCM) for feature selection, along with particle swarm optimization. The new FCM is defined as the ratio of the number of consistent patterns in the feature set. The FCM for the ith feature can be calculated as follows:

$$\mathrm{FCM}_i = \frac{number\ of\ consistent\ pattern}{number\ of\ total\ pattern} \tag{2.19}$$

Here the pattern is defined by the concept of fuzzy set theory, which is different from the traditional definitions of patterns for consistency measures. Another combination of fuzzy set theory and consistency measure can be found in (Jalali et al., 2009).

The time efficiency of consistency based feature selection algorithms is researched recently as well. In (Shin and Miyazaki, 2016), authors proposed a data set “denoising” method to eliminate examples that are seen as noises from a data set until it becomes to include consistent feature sets and then apply the binary measure to find an appropriate feature set that is consistent. Experiments show that this new algorithm outperforms in

both time efficiency and accuracy the benchmark consistency-based algorithms.

2.5.3 Correlation based Feature Subset Selection (CFS)

CFS is an algorithm that evaluate subsets of attributes rather than individual attributes (Hall and Holmes, 2003). It was first proposed by Mark Hall in his PhD thesis (Hall, 1999). In his paper, an evaluation heuristic is designed to assign high scores to subsets containing attributes that are highly correlated with the class and have low inter correlation with each other. The heuristic he uses below takes into account the usefulness of individual features for predicting the class along with the level of inter correlation among them:

$$\mathrm{Merit}_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{2.20}$$

Where Merits is the heuristic "merit" of a feature subset S containing k features, $\overline{r_{cf}}$ the average feature-class correlation, and $\overline{r_{ff}}$ the average feature inter correlation. The heuristic handles irrelevant features as they will be poor predictors of the class. Redundant attributes are discriminated against as they will be highly correlated with one or more of the other features (Hall and Holmes, 2003).

2.5.4 ReliefF

Relief was first developed by Kira and Rendell in (Kira and Rendell, 1992), and improved by Kononenko in (Kononenko, 1994), extending it with multi-class datasets and noise datasets. The main procedure of ReliefF is randomly sampling an instance from the data and then locating its nearest neighbour from the same and opposite class (Hall and Holmes, 2003). The weight of attribute is updated by each instances such that:

$$W_i = W_i - (x_i - nearHit_i)^2 + (x_i - nearMiss_i)^2 \tag{2.21}$$

Thus the weight of any given feature decrease if it differs from that feature in nearby instances of the same class more than nearby instances of the other class, and increases in the reverse case. After m iterations, divide each element of the weight vector by m and it becomes the relevance vector. Features can be ranked by their relevance vectors (Kira and Rendell, 1992).

The strengths of Relief are that it is not dependent on heuristics, requires only liner time in the number of given features and training instances, and is nose-tolerant and robust to feature interactions, as well as being applicable for binary or continuous data (Kononenko, 1994).

2.5.5 INTERACT

Zhao and Liu (Zhao and Liu, 2009) propose a feature scoring metric based on data consistency, and develop a filter algorithm named INTERACT to select relevant features while implicitly exploring feature interaction. In this algorithm, they firstly rank features according to a mutual information based measure SU, while

$$\mathrm{SU}(\mathrm{F_i}, \mathrm{Y}) = 2\left[\frac{\mathrm{M}(\mathrm{F_i}, \mathrm{Y})}{\mathrm{H}(\mathrm{F_i}) + \mathrm{H}(\mathrm{Y})}\right] \tag{2.22}$$

where *H(X)* and *H(X,Y)* denote entropy and joint entropy respectively, and *M(X,Y)=H(Y)+H(X)-H(X,Y)* be the mutual information measuring the common information shared between the two variables *X* and *Y*.

Then the consistency contribution is used to evaluate feature subsets, where

$$\mathrm{c-contribution}(\mathrm{F_i}, \mathrm{F}) = \mathrm{ICR}\left(\Pi_{\mathrm{F}-\{\mathrm{F_i}\}}(\mathrm{D})\right) - \mathrm{ICR}(\Pi_{\mathrm{F}}(\mathrm{D})) \tag{2.23}$$

where *ICR(D)* denote the inconsistency rate of set *D*. The definition of inconsistency rate is the same as in Liu's paper about consistency-based feature selection.

Test results show that INTERACT is competitive with several classical filter methods, but may stops at a very small feature subset when each class has a small number of instances (Zhao and Liu, 2009).

2.5.6 Mutual Information Feature Selection

Let *X* and *Y* be two discrete random variables. *p(x, y)* is the joint probability distribution function of *X* and *Y*, and *p(x)* and *p(y)* are the marginal probability distribution functions of *X* and *Y* respectively. The Mutual Information (MI) between *X* and *Y* can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{2.24}$$

The MI has two main properties that distinguish it from other dependency measures: first, the capacity of measuring any kind of relationship between variables; second, its invariance under space transformations.

Based on MI, a filter for feature selection was posed: Given an initial set *F* with *n* features, find subset $S \subseteq F$ with *k* features that maximizes the MI *I(C; S)* between the class variable *C*, and the subset of selected feature *S*. The Mutual Information Feature Selection (MIFS) algorithm is as follows:

Initialization: Set F as the original feature set, and set S empty.
Computation of the MI with the output class: For each $f_i \in F$, compute $I(C; f_i)$.
Selection of the first feature: Find the feature f_i that maximizes $I(C; f_i)$; set F as F\{f_i}; set S as {f_i}.
Greedy selection: Repeat until |S|=k.
Computation of the MI between variables: for all pairs (f_i, f_s) with $f_i \in F$ and $f_s \in S$, compute $I(f_i; f_s)$, if it is not yet available.
Selection of the next feature: Choose the feature $f_i \in F$ that maximizes

$$I(C; f_i) - \beta \sum_{f_s \in S} I(f_i; f_s)$$

The parameter β is a user-defined parameter that regulates the relative importance of the redundancy between the candidate feature and the set of selected features. Then set F as F\{f_i}; set S as {f_i}.
Output the set S containing the selected features

2.5.7 Minimal-Redundancy-Maximal-Relevance

Minimal-Redundancy-Maximal-Relevance (mRMR) is a multivariate ranker filter algorithm (DING and PENG, 2005). The evaluation function combines two constraints (as the name of the method indicates), maximal relevance and minimal redundancy (Bolón-Canedo et al., 2014). The former is denoted by the letter *D*, it corresponds to the mean value of all mutual information values between each feature x_i and class *c*, and has the expression shown in the following equation:

$$\mathrm{D}(\mathrm{S},\mathrm{c}) = \frac{1}{|S|}\sum_{x_i \in S} I(x_i; c)$$

(2.25)

where *S* is a set of features and $I(x_i; c)$ is the mutual information between the feature x_i, and the class *c*. The constraint of minimal redundancy is denoted by the letter *R*, and has the expression shown in the following equation:

$$\mathrm{R}(\mathrm{S}) = \frac{1}{|S|^2}\sum_{x_i, x_j \in S} I(x_i, x_j)$$

(2.26)

The evaluation function to be maximized combines these two constraints and has the expression shown in the following equation:

$$\phi(\mathrm{D},\mathrm{R}) = \mathrm{D}(\mathrm{S},\mathrm{c}) - \mathrm{R}(\mathrm{S})$$

(2.27)

In practice, this is an incremental search method that selects on each iterations and finds out the feature that maximizes the evaluation function.

2.5.8 Normalized MIFS

The Normalized Mutual Information Feature Selection (NMIFS) is based on MIFS (Estevez et al., 2009). The normalized MI between f_i andf_s, $NI(f_i; f_s)$, is defined as the MI normalized by the minimum entropy of both features

$$NI(f_i; f_s) = \frac{I(f_i; f_s)}{\min\{H(f_i), H(f_s)\}}$$

(2.28)

where $H(f_i), H(f_s)$ are entropies and *H(X)* is defined as

$$\mathrm{H}(\mathrm{X}) = -\sum_{i=1}^{n} p(x_i)\log_b p(x_i)$$

(2.29)

The average normalized MI was used as a measure of redundancy between the *i*th feature and the subset of selected features $S = \{f_s\}$, for *s=1*,…,|*S*|, i.e.,

$$\frac{1}{|S|}\sum_{f_s \in S} NI(f_i; f_s)$$

(2.30)

where |*S*| is the cardinality of set of S. This correlation measure is symmetric and takes values in {0, 1}. A value 0 indicates that feature f_i and the subset S of selected features are independent. A value 1 indicates that feature f_i is highly correlated with all features in the subset *S*. The selection criterion used in NMIFS consists in selecting the feature that maximizes the measure *G*

$$G = I(C; f_i) - \frac{1}{S}\sum_{f_s \in S} NI(f_i;\ f_s)$$

(2.31)

The NMIFS algorithm is similar with MIFS, only altering feature measure to *G*. NMIFS can reduce the bias of MI toward multivalued attributes and restricts its value to the interval {0, 1}. The NMIFS method outperformed MIFS and mRMR on several artificial data sets and benchmark problems.

2.5.9 Summary

Consistency based feature selection is famous for its quick speed and high performance. It reveals the important features by measuring consistency of features in dataset. Although many researches have enhanced this kind of filter, there is no consistency based algorithm that can provide feature ranking list. Another problem of consistency based method is that a discretization process must be applied to continuous data. The choice of discretization method will affect the performance of feature selection. As the feature ranking is vital to the hybrid feature selection process, this research will create a novel consistency based filter that can rank features according to their importance. Based on the current methods, the new method can process data with continuous values and for the problem of both binary classification and multi class reorganization. The

novel filter should also be more effective on the aspect of improving machine learning performances.

2.6 Wrappers for Feature Selection

Another category of feature selection algorithms is called wrapper which judges the quality of feature by a predetermined mining algorithm (Foroutan and Sklansky, 1985). Wrappers use the learning machine as a black box to assess feature subsets. It is suitable for the specified mining algorithm and is proved to be more effective and powerful than pure filter model. The disadvantage is also obvious, that it is very time consuming because data mining models need to be trained every time for each feature subsets.

The target of most wrappers is to find out the feature subset that has the best machine learning performance when it is used for training and validation. Basically this can be seen as a searching procedure: search all possible combinations of features from the original feature groups and find the best feature subset. An exhaustive search is only appropriate when the number of features is very small, but for large dataset with N features the size of the search space grows to $O(2^n)$. For most cases in real world machine learning problems, an exhaustively search is impractical and other searching methods are more used in wrappers.

Sequential search, include greedy hill-climbing approach, only search part of the subset groups and can only get a local best solution. Random search is also efficiency and can help to escape from optima in search space. Heuristic search strategies are also widely used in feature selection, such as GA and PSO (Liu and Yu, 2005). In this section, some benchmark searching methods of wrappers will be reviewed.

Apart from the searching strategy, to design a wrapper one also needs to consider:

(1) how to design the fitness function and
(2) which predictor to use

As ANN has been set as the predictor, there is only one other problem need to consider: how to design the fitness function. The accuracy of predictor is often used with few doubts and this thesis designs a fitness function based on this. So in this section, wrappers designed for MLP will be researched after the literature review of feature

searching strategies, along with some other benchmark wrappers with different learning machines.

2.6.1 Sequential Search Algorithms

Since an exhaustive search of the feature space is much too time consuming for wrappers, some simple searching strategies are utilized such as Greedy searching algorithm. Hill climbing searching algorithm, as an effective application of Greedy searching strategies, has obviously computationally advantages (Guyon and Elisseeff, 2003). Generally they can be divided into two categories: forward selection and backward selection.

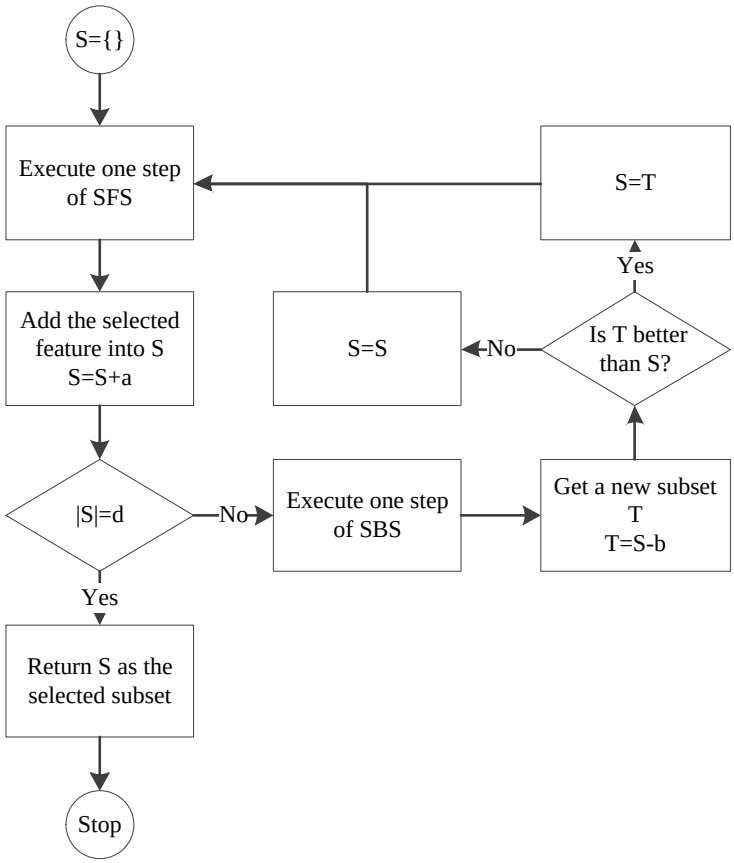

Figure 5 SFFS flow chart

In forward selection, the searching procedure starts from an empty subset and adds into features that can enhance the overall performance of learning machine. One simple example of forward searching wrapper algorithm is the Sequential Feature Selection (SFS) that was proposed by (Pudil et al., 1994). For each step, SFS adds into one feature that has the most significant improvement of the learning machine's prediction accuracy.

It stops when the size of feature subsets meets the expected number of selected features. This searching process is naïve and can be seen as a Hill Climbing search, as it ignores the dependency between features. An optimization of SFS wrapper is the Sequential Floating Forward Selection (SFFS). Comparing with SFS, SFFS contains a backtracking step that excludes a feature from the selected subset if the shrink subset has higher performance. The flow chart of SFFS is shown in Figure 5, where S is the selected feature subset and is empty at the start. *d* is the expected size of selected subsets. For each step, SFFS adds into one feature by applying SFS which is based on the evaluation function of feature subsets. Then the increased subset S applies one step of SBS which exclude one feature from S if this removing can increase the overall performance of the reduced feature subsets. The best subset is passed into the next loop or returned as the selected feature subsets if its size meets the desired size *d*.

The backward selection, on the contrary, starts with the set of all features and eliminates the least promising feature at a time. Comparing with forward selection, the backward selection is much more computationally expansive, as building learning machines with fewer features is much faster (Kohavi and John, 1997). Although the backward selection can keep dependency feature pairs remaining in the subsets, both forward and backward selections have nested subset problems. This means some correlated features may be included into the subset if they both highly enhance the performance in the evaluation. These correlated features are redundant and not necessary as smaller feature subsets are always expected.

To overcome the nested subset problem and save computing time, some optimized forward searching methods for wrappers are proposed. In (Somol et al., 1999) authors proposed the Adaptive Sequential Forward Floating Selection (ASFFS) algorithm that can produce a less redundant feature subset than SFFS. ASFFS may add more than one feature at one time, and this number is calculated adaptively. It also involves a backtracking step that is similar with SFFS. Except from wrappers, ASFFS is utilized in filters with some statistical measure based fitness functions which evaluate feature subsets without building learning machines (Sun et al., 2005).

Some other algorithms also aim at solving nesting problem, such as the Plus-L-Minus-r search (Nakariyakul and Casasent, 2009) which adds L features and remove r features in

each step. It also tries to replace one feature in the subset in each step to form a new and better subset.

2.6.2 Heuristic Search Algorithm

Genetic algorithm (GA), or sometimes called genetic programming, was first proposed in 1970s by (Holland, 1975). As a kind of evolutionary learning technique (Espejo et al., 2010), it is a high efficiency search method based on principles of natural selection and genetics (Cantú-Paz, 1998). A literature review of GA will be given later in section 2.8.

GA is now used to find the feature subset that has the best performance in machine learning. The predictors' performances are considered in the design of fitness function in GA, sometimes go along with the number of selected features. Comparing with sequential search algorithms, GA has larger searching scope and can avoid nested subset problem if given proper settings. The limitation of GA in feature selection is the possibility of falling into local optimum and the computing complexity. Parameter setting is also a problem as the settings are different for different datasets.

The algorithm of a typical GA based wrapper can be described as follows:

Initialization: create the first generation in which each chromosome represents a feature subset
while (stop criteria not meet)
    evaluate: get the fitness of each chromosome (feature subset) by fitness function
    select: select out the parents for the next generation
    crossover: generate new chromosomes (new feature subsets) for the next generation
    mutate: mutate chromosome according to the mutation rates
return: the chromosome (feature subset) with highest (or lowest, according to the fitness function) fitness

In feature selection problem, GA is used to search for a group of features that can represent the whole feature set. Aims of this task may vary from pursuing higher machine learning accuracies, eliminating irrelevant and redundant features, or finding out determinative facts of one problem. Feature selection is often treated as a multiple optimums problem. In most occasions there is no "best solution", but with some "acceptable" ones. For example, to improve the performance of machine learning, both prediction accuracies and number of features should be considered in feature selection. It will be difficult to tell which is better when improving 0.1% accuracy by adding 5

more attributes. Sometimes two or more sets of features can achieve very similar machine learning performances. GA can solve multiple optimum problems because of its natural structure. Another reason of using GA is that the feature selection problem has an exponential search space. In many pioneering researches, GA outperforms other feature selection algorithms (Brill et al., 1992, Kuncheva and Jain, 1999, Raymer et al., 2000).

To utilize GA for feature selection problem, the first thing should be done is defining the problem. In this section, three parts in GA that are specially designed for feature selection problem will be explained in detail

1. Encoding

   In this research the binary bit string is chosen as the representation of chromosome. This is specially suits for feature selection, as a bit of "1" implies the corresponding feature is selected and "0" means excluded. The length of the chromosome is denoted here as n, equal to the number of features in original datasets. The encoding representation of features is described in Figure 6.

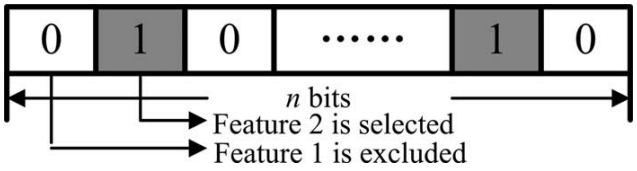

Figure 6 Encoding representations for feature selection

2. Fitness function

   To pursuing higher prediction accuracy, the fitness function in feature selection problem is always relevant to the accuracy rate or error rate of one specified machine learning algorithm applied on the candidate feature subset. For example, in a wrapper for MLP classification, the fitness score of a chromosome has positive correlation with the classification accuracy generated by the MLP that trains with selected features (indicated by 1 in the chromosome). A general expression of fitness function for chromosome c is as follows:

$$fitness(c) = P(A_c) - Pe(A_c) \tag{2.32}$$

where A_c is the corresponding feature set of c, and $P(A_c)$ is the prediction accuracy of specified learning algorithm. $Pe(A_c)$ is a penalty for A_c, which always relevant to the number of features in c. An example could be

$$Pe(A_c) = w(|A_c| - d) \tag{2.33}$$

where $|A_c|$ is the number of attributes selected in c, and d is the desired number of selected features.

It should be notice that in some SGAs for feature selection problems, neither learning algorithm nor the training process is involved. The fitness function could be the sum of mutual information, or any statistic value that can evaluate the selected feature subset (Tsai et al., 2013). These SGA algorithms are treated as filters, but not wrappers.

3. Stop Criterion

   For SGA reaching maximum generation is a typical stopping criterion in most cases. Other stop criteria include reaching a predefined fitness level, or finding the same best chromosome in *n* continuously generations.

A CHC GA, which is an optimized version of GA, was proposed in (Cordón et al., 2006) for the feature selection task in image registration. It optimized traditional GA to fit for feature selection in the following aspects:

1. Several best chromosomes are selected as parents in each generation. This makes sure the best features can be kept in the new subsets.
2. Modified crossover method makes sure half of the non-matching alleles are involved.
3. In the crossover step, only parents with enough diversity can generate the next chromosomes. This diversity is calculated by a distance function and it must be larger than the threshold *d*. The threshold is initially set to a quarter of the length of the chromosome, and decreases by 1 if there is no offspring in the next generation. This ensures a fast converge as the size of population shrinks.
4. Mutation only happens when the size of *d* drops down to zero.

The CHC GA is faster than traditional GA on wrappers as it can converge in shorter time. Mutation step also avoids being trapped into local optimum and ensures larger searching scopes.

GA based wrappers have been widely used. A multi objective GA is designed for feature selection in handwritten digit string recognition in (Oliveira et al., 2003). In paper (Kudo and Sklansky, 2000), authors compared GA with several feature selection methods, including both filters and wrappers. Test results show that SFFS fits for small and medium size datasets and GA is more suitable for large scale problems (with more than 50 features).

Besides GA, some other heuristic search algorithms are also used in wrappers. In (Tu et al., 2007) a PSO-SVM wrapper was designed by using particle swarm optimization in wrappers for the searching of subsets. According to the experiment results, this wrapper can enhance the prediction accuracy of SVM and has better performance comparing with other feature selection methods. This PSO-SVM wrapper was applied to cancer classification in (Alba et al., 2007) and was optimized to Geometric PSO in that paper. Experiments proved G-PSO and GA have competitive performances on cancer dataset.

A random searching strategy, realized by randomly remove irrelevant variables according to the computed schedule, performed high learning speed (Stracuzzi and Utgoff, 2004). It was also proved to be as capable as complete search method by experiment on decision tree and naive Bayes evaluation algorithm.

Although with higher enhancement of machine learning performances, wrappers are always time consuming because of the large amount of predictor training and validation. For each candidate feature subset, a training process must be done and along with tests to access the performance. This process will be even longer if the subset contains large numbers of features. This shortage is often covered by accelerating the converge process of heuristic search algorithms. In this thesis, the converging process of GA is accelerated by using filter’s results.

Another problem of GA based wrapper is the overfitting problem, which is caused by training data too much times. Overfitting can make the predictor has low generalization performances although with high test performances. This problem can be avoided by a

better separating of test, validation and training dataset (Kohavi and John, 1997). In this thesis, a novel data preparing method will be proposed and utilized.

2.6.3 Wrappers for ANN

To evaluate the saliency of a feature in MLP, there are mainly two kinds of measurements. The partial derivative based saliency measure calculates each feature's effect on a MLP's output by summing the partial derivatives of the outputs to that feature. The weight based saliency measure sums the squared values of the weights connecting feature *i* to the hidden nodes.

A combination of them was presented in (Setiono and Huan, 1997). The training process was minimizing the cross-entropy function:

$$F(w,v) = -(\sum_{i=1}^{k}\sum_{p=1}^{C} t_p^i \log S_p^i + (1-t_p^i)\log(1-S_p^i))$$

(2.34)

Where

k is the number of patterns

t_p^i=0 or 1 is the target value for pattern xi at output unit *p*, *p=1, 2, …, C.*

C is the number of output units.

S_p^i is the output of the network at unit *p*:

$$S_p^i = \sigma(\sum_{m=1}^{h} \delta((x^i)^T w^m) v_p^m$$

(2.35)

x_i is an n-dimensional input pattern, *i=1, 2, …, k.*

w^m is an n-dimensional vector of weights for the arcs connecting the input layer and the m-th hidden unit, *m=1, 2, …, h.*

v^m is a C-dimensional vector for the arc connecting the *m*th hidden unit and the output layer.

The output unit activation function is the sigmoid function $\sigma(y) = 1/(1+e\text{-}y)$.

The hidden unit activation function is the hyperbolic tangent function

$\delta(y) = (e^y - e^{-y})/(e^y + e^{-y})$

To find the smallest subset of features while the MLP performance remains, a penalty function was introduced to make sure those connections from the necessary inputs have large magnitude. The penalty function is denoted as:

$$f(w) = \frac{\epsilon_1 \beta w^2}{1 + \beta w^2} + \epsilon_2 w^2 \tag{2.36}$$

Thus, the function that should be minimized during training is as follows:

$$\begin{aligned} \theta(w, v) &= F(w, v) + P(w) \\ &= -\left(\sum_{i=1}^{k} \sum_{p=1}^{C} t_p^i \log S_p^i + \left(1 - t_p^i\right) \log\left(1 - S_p^i\right) \right) \\ &+ \epsilon_1 \left(\sum_{m=1}^{h} \sum_{l=1}^{n} \frac{\beta (w_l^m)^2}{1 + \beta (w_l^m)^2} \right) + \epsilon_2 \left(\sum_{m=1}^{h} \sum_{l=1}^{n} (w_l^m)^2 \right) \end{aligned} \tag{2.37}$$

Based on this augmented evaluation function, the *Neural-Network Feature Selection Algorithm* described in papers is as follows:

(1) Let $A=\{a_1, a_2, \ldots, a_n\}$ be the set of all input attributes. Separate the patterns into two sets: the training set S_1 and the cross-validation set S_2. Let $\triangle R$ be the allowable maximum decrease in accuracy rate on the set S_2 and let $\epsilon_1(\mathrm{k})$ and $\epsilon_2(k)$ be the penalty parameters for the connections from input A_k to the hidden layer, for all $k=1, 2, \ldots, n$.

(2) Train network N to minimize $\theta(w, v)$ with the set A as input such that it achieves a minimum required accuracy rate on the set S_1. Let R^2 be the accuracy of the network on the set S_2.

(3) For all $k=1, 2, \ldots, n$, let N_k be the network whose weights are set as follows.

From all inputs except for A_k, set the weights of N_k equal to the weights of N.

Set the weights from input A_k to zero.

Compute $\mathrm{R}_{\mathrm{k}}^1$ and $\mathrm{R}_{\mathrm{k}}^2$, the accuracy rates of network N_k on the sets $S1$ and $S2$, respectively.

(4) Rank the networks N_k according to their accuracy rates. Let $\mathrm{R}_{\mathrm{ave}}^1$ be the average of these rates.

Set $k=1$.

Retrain the network $N_{r(k)}$.

Let $\delta = (R^2 - R^2_{r(k)})/R^2$.

If $\delta \leq \triangle R$, then

Update the penalty parameters for all attributes $j \neq r(k)$:

For each input attributes a_j with network accuracy rate $R^1_j \geq R^1_{ave}$, set $\epsilon_1(j) = 1.1\epsilon_1(j)$ and $\epsilon_2(j) = 1.1\epsilon_2(j)$

For each input attributes a_j with network accuracy rate $R^1_j < R^1_{ave}$, set $\epsilon_1(j) = \epsilon_1(j)/1.1$ and $\epsilon_2(j) = 1.1\epsilon_2(j)/1.1$

Reset the input attribute set to $A-\{a_{r(k)}\}$, and set $N=N-1$, $N=N_{r(k)}$.

Set $R^2=max\{R^2, R^2_{r(k)}\}$.

Go to step 3.

If $k<N$, set $k=k+1$ and go to step 4b.

Else stop.

In this algorithm, variety of both output accuracy and connection weights were taken into account of measuring features. This representative wrapper for MLP has high computing efficiency by adding penalty function and can successfully remove irrelevant features. However, it could not eliminate redundant features as they can have larger connection weights.

2.6.4 Summary

GA has short searching time and has been proved to be fit for wrappers. However, there are still shortages of GA wrapper when applying on ANN: slow converging speed and unpredictable initialization. Also, there are very few wrappers that are designed for ANN, as the training of ANN is too long. To cover these shortages and adapt to hybrid feature selection process, this research proposed a novel GA based wrapper for ANN, with short processing time and obvious improvement of prediction accuracy.

2.7 Hybrid Feature Selection

Recently some hybrid models showed high performance on feature selection. Many experiments proved that wrappers are more effective in the aspect of improving data mining models generalization abilities, while filters take less computing time (Estevez et al., 2009). From this point, combining filters and wrappers is designed to achieve a fast and effective feature selection on high dimensional datasets.

Basically, there are two ways to merge filters and wrappers. One solution is a liner structure: apply filters before wrappers. The advantage of this method is that candidate

features are first selected from the original datasets via computationally-efficient filters. Then the selected feature groups are further refined by more accurate wrappers. Another way is to embed filters into the searching strategies of wrappers to achieve a fast locating of desired feature subsets. This method is more suitable for models with long training time, for example, BP training algorithms for ANN. In this section, literatures for both kinds of methods will be reviewed, along with the embedded feature selection methods which is sometimes categorised as a separated kind of feature selection algorithm.

2.7.1 Use Filters Inside Wrappers

Hybrid Genetic Algorithm (HGA) is a typical way of using filter inside wrapper for accelerating the computing speed. It has been developed to overcome the limitation of GA, but not just for feature selection. There are three ways of hybridizing GA: problem-specific encoding, the use of special genetic operators, and the incorporation of the good features of classical algorithms (Davis, 1991). HGAs have shown outstanding performance on some research areas (Bui and Moon, 1996, Jog et al., 1989, Zheng et al., 1997).

In (Oh et al., 2004), a hybrid genetic algorithm (HGA) is proposed and shown good performance on 1-NN learning algorithm. It can quickly converge to local optimum by using “add” and “del” local search operators. Each solution in each generation will add or delete features until finding the best local optimum and then goes into the next generation. The local search operators (LSO) are defined as:

rem^{g}: remove the least significant g features which can maximize $J(c)$

add^{g}: add the most significant g features which can maximize $J(c)$

where J(c) is either the accuracy of a specific classifier on chromosome c, or a generic statistical measurement (e.g., the filter approach). Although only the LSO with classification accuracy is tested in that research, it can still be seen as proposing a new method of hybrid feature selection.

(Huang et al., 2006) improved it by adding mutual information into the fitness function of GA. For every feature subset generated in each generation, conditional mutual

information that measures the contribution of new information to the output class C offered by feature f_i given the subset S of features selected is calculated. Local search measure is defined as:

$$I(C; f_i|S) = I(C; f_i) - \frac{1 + \beta log_2^{|S|}}{|S|} \sum_{f_S \in S} \frac{I(C; f_i)}{H(f_S)} I(f_S, f_i) \quad (2.38)$$

where $I(C; f_i)$ is the mutual information of feature *C* and f_i.

This method is especially fit for hybrid feature selection, as in paper (Zhu et al., 2007) feature ranking list replaces the local optimum searching, which combines filters into wrappers. The paper improved the HGA method by using the result of filter approach instead of calculating *J(c)*. The LSOs add or delete one feature at a time, and are defined as:

add: add the highest ranked unselected feature into the selected feature set

del: delete the lowest ranked selected feature

The most significant difference from the previous work (Oh et al., 2004) is the local searching strategies. Two different searching strategies are designed and tested in paper (Zhu et al., 2007): improvement first and greedy. In improvement first strategy, add and del operators are both applied random times. The search stops at the first improvement of classification accuracy or reduction of features. Greedy search strategy evaluates all possible combinations of add and *del* operations, and replace the original chromosome with the best one. The algorithm of improvement first strategies is as follows:

Improvement first strategy algorithm:
Initialize l and w
For each chromosome c among the w elitists
    For (j from 1 to l^2)
        Generate a unique random pair (k,d)
        Repeat k times of add operation
        Repeat d times of del operation
        If F(c')>F(c) or (F(c) – F(c'))<ε and |c'| < |c|)

Replace c with c'
Break and consider the next elite chromosome
End

where *l* is the local search length and *w* is the interval. The settings of them in their tests are *l* = 2, 4, 8 and *w* = 1, 5. *F(c)* denotes the fitness score of chromosome *c* and in their research it has

$$F(c) = J(c) \tag{2.39}$$

which means the fitness score is equal to the classification accuracy of the specified classifier. The only difference of greedy strategy is searching l^2 combinations of k and d and replace c with the best c'.

As the fitness function have no restriction of the number of selected features, this hybrid feature selection algorithm may tend to evaluate large feature sets. To constrain this number to a maximum of m, restrictive crossover and restrictive mutation process are designed in their research. The value m is defined by prior knowledge of the specified data, and in their experiments it is set to the size of original datasets for small and medium datasets, and 50 for large datasets. Three different feature ranking methods are tested: ReliefF, gain ratio and chi-square. The overall hybrid feature selection algorithm is as follows:

WFFSA algorithm
Initialize with SGA initialization algorithm
While stop criteria are not satisfied
Evaluate all chromosomes
For each of the best w chromosomes
Perform local search and replace with the improved chromosome
Perform SGA selection operator
Perform restricted crossover
Perform restricted mutation
End

Advantages of WFFSA include a faster searching speed compared with the HGA in paper (Oh et al., 2004) and SGA for feature selection. Also it proves that the

combination of wrappers and filters can achieve high efficiency and effectiveness. However, similar with all GA, it has lots of parameters to be set by prior knowledge. Secondly, as only the best or worst feature will be added or deleted, some middle ranked features may have very little chance to be altered, especially for large datasets. Another limitation is that WFFSA still needs 6000 function calls for small and medium datasets, which could be intolerable for MLP feature selection.

A hybrid GA was used for feature selection in (Kabir et al., 2011) called HGAFS. It incorporated a local search operation that was devised and embedded in HGA to fine-tune the search in FS process. The local search technique worked on basis of the distinct and informative nature of input features that is computed by their correlation information.

Recently a hierarchical micro GA was proved to have high efficiency when training and perform well in result validation. It used neural networks as mining algorithm and solved a real world human motion detection and classification problem (Tan et al., 2014).

2.7.2 Use Filters Before Wrappers

Another kind of hybrid feature selection methods is to use filters before wrappers which ensure fast feature subset searching for high dimensional datasets. Filters are used to dramatically reduce feature amounts and wrappers are for enhancing the performance. One typical solution is Re-ranking based Feature Subset Selection (Bermejo et al., 2012). It can dynamically decide the amount of features that should be kept in filter process, which ensured all important features being inspected in wrappers while eliminating lots of irrelevant features. The algorithm is as follows:

```
list R = {} // The ranking, best attributes first
for each predictive attribute Ai in T
        Score=MT(Ai, class)
        insert Ai in R according to Score
sol.S = ∅ // selected variables
sol.eval = null // data about the wrapper evaluation of sol.S
B = first block of size B in R // B is ordered
remove first B variables from R
```

```
sol = IncrementalSelection(T,B,C,S)
continue = true
while continue do
        R′ = {}
        for each predictive attribute Ai in R
                Score=MT(Ai, class|S)
                insert Ai in R′ according to Score
        R = R′
        B = first block of size B in R // B is ordered
        remove first B variables from R
        sol′ = IncrementalSelection(T,B,C,S)
        if(sol.S == sol′.S) //no new feature selected
                then continue = false
        else sol = sol′
return (sol.S)
```

The main idea of this algorithm is that better variables are ranked in the first positions and so variables having a worse ranking can be directly discarded. This idea is also used for example in BARS (Ruiz et al., 2008), where the authors recommend that only the first half of the ranking should be explored. Another representative algorithm exploiting this idea is LFS (Gutlein et al., 2009), where only a small portion of the rank is explored, e.g the first 100 variables.

One early research about hybrid feature selection is given by Sanmay Das in paper (Das, 2001). The proposed algorithm BDHFS (Boosted Decision Hybrid Feature Selection) uses boosting and incorporates some of the features of wrapper methods into a fast filter method for feature selection. Boosting assigns higher weights to examples that have often been misclassified in the previous rounds. In each step, feature with highest weighted information gain is added into the training group. If the training accuracy stops increasing, the search stops without adding the last feature. K-NN, Naïve Bayes network and ID3 classifier are used in the experiments. The results show that BBHFS improves the performance of the learning algorithm significantly in most cases and is competitive with wrapper methods while selecting features much faster. The drawback of this algorithm is that the number of selected features needs to be predetermined.

More recently, Cadenas et. Al (Cadenas et al., 2013) proposed a hybrid feature selection method to handle low quality data which contain imprecise and uncertain instances. The proposed algorithm (FRF-fs) is based on a Fuzzy Random Forest, with a sequential search procedure. It includes three steps:

(1) Scaling and discretization process of the feature set; and feature pre-selection using the discretization process;
(2) Feature subset pre-selection with FRF ensemble, and rank features with an FDT and information gain based process; and
(3) Wrapper feature selection using a classification technique based on cross-validation.

In the wrapper step, features that can increase accuracies are put into one group. Then based on this feature group, a greedy search procedure selects out the fewest features that can achieve highest accuracy. Experiment results show that feature subsets selected by this method have good performance with both regular or low quality datasets. But the searching scope of this method is restricted.

Ant Colony Optimization (ACO) is used in searching of better feature subsets as it can get one good result with only few subset evaluations (Shahzad, 2012). Mutual information of each feature is calculated first. Then an ACO process is applied with the fitness function:

$$Merit(S) = \frac{[N - s]\sum_{i=1}^{N} |c_{ic}| - \alpha \sum_{i=1}^{N}\sum_{j=1}^{N} |c_{ij}|}{N} \quad (2.40)$$

Where S is a constructed solution comprised of a local memory of an ant, N is total number of features in a dataset, s is the number of features in solution S, $|c_{ic}|$ is the correlation between a feature i and the class c. $|c_{ij}|$ is the conditional mutual information. 13 datasets are used in the experiment, with the largest dataset contains 69 features. The results show that this algorithm has good performance on KNN and SVM classifiers. However, performance with larger datasets is expected to be tested.

2.7.3 Embedded Methods

Embedded methods try to incorporate feature selection process as a part of the training of learning machines (Guyon and Elisseeff, 2003). The aim is to reduce the computation time which is very long in wrappers, and has better performances than filters (Blum and Langley, 1997). Although in some papers the embedded feature selection methods are defined as a separate category, this work considered it as one kind of hybrid feature selection methods because most embedded methods contains both filters and wrappers.

For example, in (Hanchuan et al., 2005) a two stage feature selection algorithm is designed by combining mRMR with wrappers. In the first stage the mRMR filter method is used to calculate *k* which is the optimal number of features, and then the second stage uses wrappers to find out the best subset that has *k* features, according to the prediction accuracies. This method is considered as a kind of embedded methods in (Chandrashekar and Sahin, 2014).

Another way to remove irrelevant features is by investigating the trained predictor, like the weights of the first layer in a MLP. In paper (Guyon et al., 2002) a novel embedded feature selection method Recursive Feature Elimination (RFE) was designed to select a small subset of genes from DNA micro-arrays. It used the definition of coefficient which was proposed in (Golub et al., 1999) as follows:

$$w_i = (\mu_i(+) - \mu_i(-))/(\sigma_i(+) + \sigma_i(-)) \tag{2.41}$$

Where μ_i and σ_i are the mean and standard deviation of the samples in class (+) and class (-). This weight w_i for each feature stands for the correlation or anit-correlation with the target class (Guyon et al., 2002). Hence it can be used to define an imperfect predictor by the feature itself, and forms a voting scheme as:

$$\mathrm{D(x)} = \mathrm{w} \cdot (\mathrm{x} - \mu) \tag{2.42}$$

where μ is the mean of the data. Thus the correlation measure *w* can be used as weights in predictors. Features with smallest *w* will be eliminated by setting its weight to zero.

This kind of embedded feature selection methods is applied to SVM predictors in (Guyon et al., 2002, Mundra and Rajapakse, 2010) and known as the SVM-RFE algorithm. In (Chapelle and Keerthi, 2008) it was modified to be fit for multi-class problems.

Similar concept of adjusting weights in predictors is also applied on ANNs. This kind of algorithm is often named Network Pruning, i.e, apply penalties to features with small weights and exclude the nodes connected to these inputs (Setiono and Liu, 1997, Romero and Sopena, 2008).

2.7.4 Summary

Some researches get the same conclusion that hybrid feature selection can enhance the performance of predetermined data mining algorithm with higher accuracy than filters, and less computing time than wrappers (Sarafrazi and Nezamabadi-pour, 2013). But this is only obvious to datasets with large number of attributes (Zhu et al., 2007). The details of one hybrid feature selection method (Oh et al., 2004) will be introduced in chapter 6, as the basic idea of new method in this thesis is based on that.

2.8 Genetic Algorithm

Genetic Algorithm (GA) is a kind of Evolution Algorithm that searches the best solutions in the problem scope. It is a search heuristic that mimics the process of natural selection. The group of solutions is called “population” in GA, and each candidate solution is encoded as “chromosome” (or genotype). Similar as the biological gene, chromosomes can crossover, mutate and generate new chromosomes, which form new generations of population. Following the idea of “survival of the fittest” which was proposed by Charles Darwin in 1859, GA process is designed to optimize solutions that can “better” solve the problem. To evaluate the fitness of each solution, fitness functions should be designed in GAs. The fitness score of each chromosome represents its ability of solving the problem, and chromosomes with higher fitness scores have higher chance of generating new generations, which implements “survival of the fittest”. The search process stops when the whole population becomes stable, or when it meets predefined stop criterion. A typical GA is described in Figure 7.

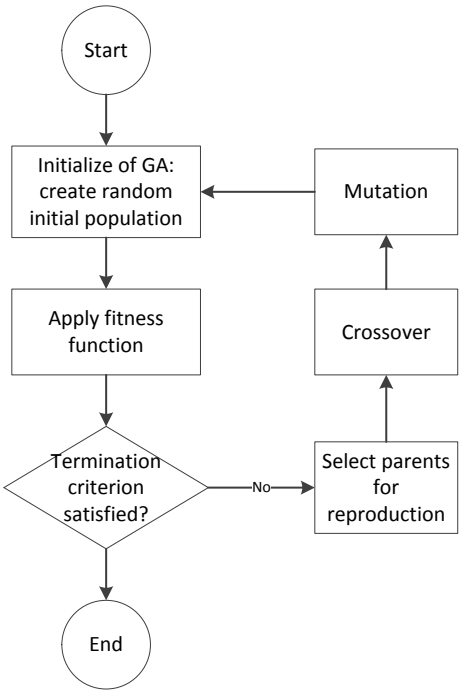

Figure 7 Typical Genetic Algorithm

It usually starts from initializing parameters and creating an initial population randomly. This population is the first generation and the amount of it is determined by experience. Then a fitness function is applied on each solution to calculate the fitness value respectively. If some solutions with fitness values that meet the stopping criterion, the algorithm stops. Otherwise, parents of next generation are selected according to fitness value and generate solutions by crossover and mutation methods. This iteration goes on until termination condition is reached. As GA will be optimized in this thesis, details of its implementation will be described in this section.

2.8.1 Encoding

GA simulates the biological evolution of genetics. As such it is consisted of population and chromosomes. To represent a solution in a way of computer program language, a string of real numbers or more typically a binary bit string is used in GA. This string is usually called chromosome in GA, and may looks like this:

10001110101100100

In the first step of GA, the first population is created and solutions (chromosomes) are initialized as binary bit strings. The population may look like this

Chromosome 1: 10111010010100

Chromosome 2: 01101000101010
Chromosome 3: 10101110101010
Chromosome ……

2.8.2 Fitness Function

Fitness function is designed to calculate the fitness of each chromosome. It is defined according to the specific problems. It has huge effect to GA, as parents of new chromosomes are selected based on their fitness value: chromosomes with higher fitness are more likely to be selected.

In some research, the number of calling fitness function is used to present the efficient of GA. A bookkeeping method is designed for GA to reduce calling of fitness function. As some chromosomes may appear more than one times, it will be more efficient to bookkeep the already evaluated chromosomes, which can avoid duplicate computations and speed up the algorithm.

2.8.3 GA Operators

To mimic biological evolution, three operators are used in GA to generate new chromosomes from the old one: select, crossover and mutate.

2.8.3.1 Select

There are many different ways of selecting chromosomes. All of them are based on fitness function scores of chromosomes (Marczyk, 2004). Roulette Wheel is a classical way and Elitist selection can keep the best chromosome into next generation. A combination of them with Scaled Selection will be used in this research. Only an overwhelming description will be presented here and the implementations of them will be in section 6.2.

1. Roulette Wheel

Roulette Wheel selection is utilized in the standard GA (SGA) and is most similar with biological natural selection. Imagine that each chromosome is represented in a pie chart (the Roulette Wheel, in Figure 8) as a slice. The size of the slice is proportional to that chromosomes fitness score and the total chart is the sum of all fitness. The wheel is then spun, and whichever individual "owns" the section on which it lands each time is chosen.

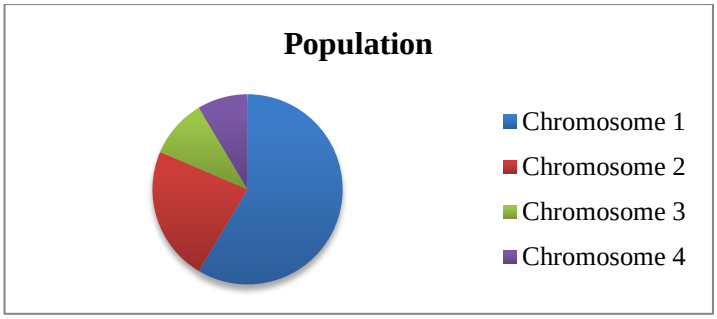

Figure 8 Roulette Wheel selection in GA

In Roulette Wheel selection, the "best" chromosome that has the highest fitness may not be chosen. However, it has very high chance of going into next generation.

2. Elitist Selection

Different from Roulette Wheel selection, elitist selection can guarantee that the best chromosome will be selected and copied into the next generation. This algorithm implements "best survive" in biology. Elitist selection, according to some researches, can obviously shorten the GA process, but may tend to trap in local optimism. Modified elitist selections, in which the single best or a few of the best are selected, are always preferred, which may achieve quick converge and large searching scope.

3. Scaling Selection

Scaling selection is preferred when one or few extreme large fitness scores in one generation appear in the beginning. Also, when the average fitness of the population increases, the strength of the selective pressure increases and the fitness function becomes more discriminating (Marczyk, 2004). Scaling the fitness scores can maintain an even selection pressure throughout the GA. There are three common categories of scaling function: linear scaling, rank scaling, exponential scaling and top scaling (Sadjadi, 2004). The scaling functions of these four are as follows:

$$Linear\ Scaling: f_{linear} = a + bf_{raw} \tag{2.43}$$

$$Rank\ Scaling: f_{rank} = p - 2\frac{(r-1)(p-1)}{N-1} \tag{2.44}$$

$$Exponential\ Scaling: f_{exponential} = m^{r-1} \tag{2.45}$$

$$Top\ Scaling: f_{top} = \begin{cases} sN, & \text{for } r \geq c \\ 0, & \text{for } r < c \end{cases} \tag{2.46}$$

where r is the "rank" of the chromosome, p is the desired selection pressure (best/median ratio), N is the size of the population, m is the times that new fitness is greater than the previous ones, s is the proportionality constant and c is the number of chromosomes that will be scaled up.

In this research, linear scaling will be used and the details will be introduced in Section 6.2.4.

4. Other Selection

Other selection methods include fitness-proportionate selection, tournament selection, generational selection, steady-state selection and hierarchical selection. All these selection methods have different advantages and limitations. Combinations of them are preferred in various situations.

2.8.3.2 Crossover

Crossover operator is the prime factor that distinguishes GA from other searching methods. After the choosing process, no new solutions are generated because all chromosomes remain same as the previous generation. Crossover can generate new chromosomes and at the mean time keep the gene set unchanged. That means, crossover will not introduce new gene but can produce new chromosomes, which can enlarge searching scope.

A typical crossover operator starts with selecting two chromosomes by selecting operator. Then according to a predefined crossover rate (usually around 0.7), the two chromosomes are copied into the new generation or go into the crossover process.

There are many ways of crossover that fits for different data structures, including one-point crossover, two-point crossover, "cut and splice" and uniform crossover. This

research uses the most simple and common one-point crossover. Two chosen chromosomes swap their binary bits after a single crossover point. The position of this crossover point is usually selected by random. The process of one-point crossover is described in Figure 9.

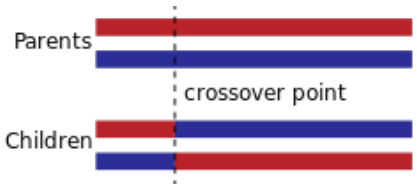

Figure 9 One point crossover

2.8.3.3 Mutation

Mutation is an important way of creating new genes. Simple mutation randomly flips bits in selected chromosomes. It is always performed after crossover, and takes place with a random chance at a mutation rate (usually very low, around 0.001) (Chandrashekar and Sahin, 2014).

2.8.4 Stop Criterion

The GA process stops if it reaches the maximum number of generations. There are also many other stop criteria for different searching problems. For example, if the fitness of a chromosome is larger than a predefined value, then this chromosome can be seen as the desired solution so GA can stop. Or in some cases GA can stop when the best fitness score remains same for n generations.

2.8.5 Summary

One significant advantage of GA is its highly parallel structure. As multiple chromosomes are evaluated in each generation, they can explore the solution space in multiple directions at once. Due to its parallel searching process, GA is well suited to solving problems where the space of all potential solutions is huge. These problems are usually too large to search exhaustively considering reasonable amount of computing time. Another strength is that GA performs well when the fitness function is complex: discontinuous, noisy or changes over time. The mutation of chromosome can avoid GA trapping into local optimum. However, it should be noticed that there is no guarantee finding the global optimum in GA, while it can always find acceptable solutions by only evaluating a set of them. This advantage also makes GA suitable for multi objects

searching problems. At last, GA use no previously known domain-specific information to guide the search, which means it will not be biased towards any searching directions. This may make the searching process longer though, but the advantage is this random change to solutions ensures an open searching scope.

Although GA has large searching scale and quick speed to get a good solution, it has some drawbacks. First, as the fitness function is vital in GA, the definition of it must be carefully considered. For some complex problems which the desired solutions should have multiple features, an imprecisely chosen fitness function may make GA unable to find a good solution. Also the other parameters in GA, such as the mutation rate, crossover rate, size of population and others, must be defined carefully. There is no best choice or equations to calculate their values, but a poorly chosen of them may bring disaster. Another problem of GA is the “deceptive” fitness functions, those where the locations of improved points give misleading information about where the global optimum is likely to be found. Finally a prevalent problem of GA is known as premature convergence. If a chromosome in the first generation has extremely large fitness comparing with other randomly initialized solutions, then this chromosome is highly likely to produce abundant new chromosomes in the next generation. This may result in very small searching scope and converge on the local optimum. Scaling selection can be implemented to solve this problem, along with other optimization of GA.

2.9 Summary

Although ANN can successfully solve some real world problems, when the number of attributes gets higher, the efficiency of ANN model decreases obviously (Dayhoff, 1996). Thus, applying feature selection on ANN is essential and important as it makes ANN possible to handle extremely large datasets by reducing number of input nodes. Class imbalance can also hinder the performance of feature selection and ANN.

By using feature selection methods, some input attributes can be eliminated and the number of features will be fit for ANN. The remained features are more suitable to reflect the characters of models, which ensure high generalization ability and fast training process. By now, feature selection has been applied to many fields such as text

categorization, image retrieval, customer relationship management, intrusion detection and genomic analysis (Ting et al., 2010).

However, as to ANN feature selection, there are many difficulties. Data preparing methods for class imbalance problem are either adding or removing instances, which obviously have side effect on the performance of ANN. Filters can only provide limited improvements, while wrappers have very long running time. Hybrid feature selection is a prospective way of improving the performance of ANN, but very few works have been done yet.

This research is going to use a hybrid feature selection model combining filters and wrappers to achieve an enhancement of ANN speed and generalization abilities. Instance Selection as a preliminary research has successfully solved the class imbalance problems in datasets. Details of research methodologies and designs of new algorithms will be presented in the next few chapters.

Chapter 3 - Overall Research Methodology

3.1 Introduction

The background research indicates that current feature selection algorithms have shortages such as low computing speed for ANN and no proper outputs of consistency based filters. To improve the performance of ANN by feature selection, 4 research problems should be solved:

(1) Class imbalance problems in datasets can greatly affect the performance of ANN.
(2) There is no filter specially designed for hybrid feature selection.
(3) The existing feature subset searching strategies in wrappers are not suitable to ANN.
(4) Hybrid feature selection models cannot achieve fast speed and high performance at the same time.

To solve these problems, this research designed novel algorithms and the methodologies of these solutions are provided in this chapter. The novel data preparing method based on instance selection is designed for class imbalance problem. A novel filter based on consistency based feature selection is used to solve problem 2. A novel wrapper based on Genetic Algorithms (GA) is specially designed for hybrid feature selection, and it will solve problem 3. The last problem which is also the main aim of this research will be solved by a novel hybrid feature selection algorithm. It combines the novel filter and wrapper in multiple ways.

3.2 Methodology of Designing Novel Data Preparing Method

In this research, a novel data preparing method has been developed for solving class imbalance problems in datasets. As mentioned in Chapter 2, this new method focused on rearranging the distribution of samples in training, validation and testing datasets.

One problem that caused by Class Imbalance problem is that there may be too few instances in training or testing with one class label. To avoid this situation, a new instance distribution method is designed when allocating the instances into training, validation and testing datasets. The new method ensures that every data group, training, validation and testing, can get the same proportions of instances from each class. Thus,

each data mining procedure can get a set of data which represents and remains the features of the original dataset.

However, in MLP training or other ANN approaches, training, validation and testing are repeated many times before generating a good model. The performance of a model should be the statistic results of n-time learning procedures. To descend the affection of noise data, it is necessary to generate different training, validation and testing data groups each time. Thus, the new method should also be able to select instances randomly.

3.3 Methodology of Designing Filters

Filter model applies independent evaluation criteria without involving any mining algorithm. It does not inherit any bias of a mining algorithm and it is also computationally efficient. In this research, the hybrid feature selection process needs a filter that can evaluate each feature objectively with high efficiency.

These characters make it possible to exclude highly dependent features that have little impact on the classification accuracy but waste a lot of computing time. However, most filters aim at achieving a small subset that only includes highly distinct, informative or consistent features, as the number of features directly depend computational efficiency. To be applied before wrappers, filters must perform high efficiency and effective.

As to evaluation function, consistency measure that can select out features with high consistency is used in this research. This kind of measure has been proved to be effective by Consistency-Based Feature Selection (CBFS) (Dash et al., 2000).

In this research, an enhanced CBFS is designed. The new filter is more biased on consistent features by altering the computation function of feature weights. Difference between consistent and inconsistent features becomes more obvious, which is also the requirement of the whole hybrid feature selection method.

More than that, a new measure, concentration measure, is introduced into this new filter. This measure can select out features whose values are more concentrate to each class.

The final evaluation function used in this research is called Consistency-Concentration Rate (CCR), which combines concentration and consistency character.

The performance of filters cannot be shown by their outputs. A set of features selected out by features could be meaningless and difficult to compare with other filters. Thus, it is necessary to find out an evaluation standard for filters.

In this research filter is designed to generate a ranking list of features. According to the aims of this research, the filter should enhance both the efficient and accuracy of machine learning predictions. So, the evaluation of filters should be based on the performance of predictors, which include MLP, Decision Tree and Naïve Bayes Networks.

AUC is used to assess the accuracy of MLP prediction. It can eliminate the affection of imbalanced data and at the same time objective to all kinds of training models. So it can be used in comparison with other filters. The amount of selected features is also an important evaluation of filters, as fewer features always lead to less computing time.

3.4 Methodology of Designing Wrappers

Currently few works focus on wrappers applied on ANN, especially MLP model. As ANNs are proved to have good performance on some datasets, developing a wrapper for ANN is important. Many feature selection algorithms applied on ANN use greedy algorithms as searching strategies to achieve acceptable computational complexity. However, greedy algorithms can only achieve local optimum and with long computing time. GA is suitable for random searching as it can avoid stuck into local optimum. But when the number of features is too large, it will take long time to converge and does not fit for BP training. In this research wrappers will be designed to improve MLP models in both training efficiency and classifying accuracy.

Feature subset search strategy is critical to the efficiency of feature selection. This research will develop a new search method called AEGA, which will converge quickly to local optimums and with less machine training processes.

This can be achieved by local optimization of GA. The procedure of local search in GA can be described in Figure 10.

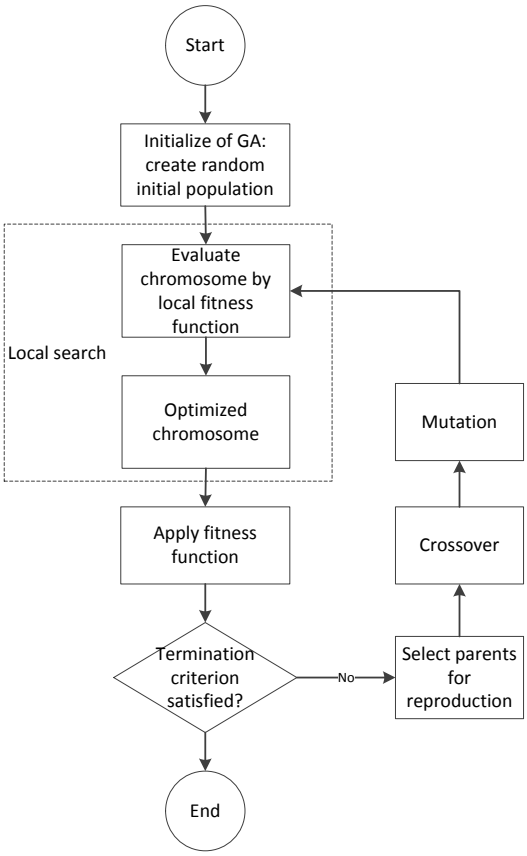

Figure 10 Local Search in GA

The local optimization can reduce generations in GA, which will save computing time. This is very important to MLP feature selection as the training of MLP takes long time.

An Elitism selection method will be applied in this research. The best solution in each generation is stored in the elitism list, for the sake of robust solution evaluation and objective stop criteria.

It should be noticed that the local search operators are designed in hybrid feature selection, as the ranking list of CCBFS will be used in this process.

Fitness function is vital for GA searching, as it directs the searching process and the outcome. The accuracy of model will be used as fitness evaluation function. However, the number of input attributes should also be taken into consideration.

Stopping criteria determines when the feature selection process should stop. In this research the process stops when a sufficiently good subset is selected evaluated by model accuracy. As ANN training contains random procedures, and 10 time validation

obviously makes the training 10 times longer, an accumulated fitness score of each solution will be used in stop criteria, along with restrictions of the maximum generation.

The goal of fitness function design is to find a good solution in the shortest time. “Good” means the wrapper may not always find the best solution, and shortest time is achieved by calling less fitness functions, as there is a machine learning process in each calling.

3.5 Methodology of Designing Hybrid Feature Selection Method

Filters and wrappers are two different models for feature selection. However, only a few works about their combination have been tested and the advantages are not obvious. This was because when considering both models, the efficiency of selection will decrease and test results are similar to filters and wrappers. In this research, as MLP training is extremely complex, it is meaningful to develop a hybrid model which can decrease training time.

The new hybrid model will use filters before wrappers to reduce amounts of features that need to be considered in wrapper process.

In this research, CCBFS provide a pre selection for wrapper, which greatly reduces the computing time. The result of CCBFS is used in the initialization of AEGA, as two solutions in the first generation. As CCBFS is a feature ranking process, the number of selected features is decided by experience. A Dataset Adapted Select (DAS) is designed to achieve a dynamic feature subset generation of CCBFS. However, this feature subset does not need to be accurate, as the searching process will automatically find the local optimum.

The filters are also used in the local search operators of GA in wrappers. The rank list of features produced by filters will be used as memes or local search heuristics. This research will follow the basic idea of WFFSA (Zexuan et al., 2007), but modifies the methods to automatically choosing parameters by inspecting filter rank list.

The new local search operators (LSO) will be designed, which utilize CCBFS as the local search pressures. In this process, local optimum does not need to be found in every chromosome, but will be achieved by further searching. This ensures short computing time and large searching scope.

3.6 Research Tools

As the tools used in research will affect the performance of ANN, especially the computing speed, both hardware and software used in this research will be introduced in this section.

This research is a purely quantitative approach, which means it has no interview, survey work or case study. It designs new algorithms that can enhance the performance of ANN, and the contributions are presented by data analysis and comparisons. So in this section, the research tools only involve the hardware equipment and software apps that are used.

3.6.1 Hardware

In this research, to avoid bias on evaluating the performance of ANN, all tests run on the same PC provided by UTAS. This PC is a Macbook Pro (13-inch, Mid 2012). It has an Intel® Core™ i5-3210 CPU, 16GB DDR3 memory (2 sticks 8GB each, 1600Mhz) and 500GB hard disk (5400rpm). The video card is not used to accelerate computing in any tests in this research. Other parts of this PC, such as outside cooling fans, keyboards and monitors have no effect on the performance evaluation in this research.

3.6.2 Software

To apply and test new algorithms, some software is necessary in this research.

Notepad++ is used to process data and transform documents into proper formats. It is a powerful free software that can process many commonly used document types. As this research download data from different websites, with different formats, this software can easily solve the data format problems.

To realise algorithms, Java programing language is used in this research. Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is one of the most widely used and most welcomed programing language in the world, and has been used to realise many machine learning and data mining algorithms. One well known machine learning software program, Weka, is also written by Java. It is used in this research and will be introduced in the following paragraph.

Weka is a collection of machine learning algorithms for data mining tasks in Java, developed by the Machine Learning Group at the University of Waikato. It has a user interface version for applying machine learning algorithms easily, and a developer version for researchers to develop new algorithms based on Weka. In this research, some algorithms are tested by Weka to give comparison with the new algorithms. The new algorithms also use some base classes (such as “instances”, “attributes” and some functions) in Weka.

Eclipse is the most widely used Java IDE (integrated development environment) and supports the development on Weka very well. In this research, it is used as the IDE and all programming and testing work is done on it.

During the experiments results collection and analysis, and the writing of this thesis, MS Office is used in this research.

All software, except MS Office, is open source and free to use.

3.7 Data Collection

This research needs data for examining the performance of new algorithms. All data are collected from Internet. Details of datasets used in this research can be found in section 7.2.

As a pure quantitative approach, this research does not concern about research ethics in interview, survey or any form of biological research. However, as the data in all datasets used here are from real world problems, such as credit accounts information, disease diagnose information and others, it is possible to contain sensitive personal information. To avoid ethic problems of using personal data, all datasets have been checked, with sensitive information such as IDs, names, addresses deleted.

To avoid copyright problems, this research only uses open source data that are published for the purpose of research. Data downloaded from two machine learning websites are used in this research: UCI (Lichman, 2013) machine learning repository and Feature Selection Dataset (FSD) (Li et al., 2016).

The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical

analysis of machine learning algorithms. It contains 351 datasets (at the time of writing this thesis) in different categories of real world problems.

scikit-feature is an open-source feature selection repository developed at Arizona State University. Its website provides about 30 datasets and most of them are widely used in feature selection research. Besides the datasets, the website also provides about 40 popular feature selection algorithms for application, research and comparative study.

3.8 Summary

In this chapter the research methodologies of each step are introduced in general. A novel instance selection based algorithm will be designed to solve class imbalance problem. A novel filter based on consistency based feature selection can provide feature ranking lists and can be utilized in hybrid feature selection process. The new wrapper is designed based on the idea of GA and the hybrid feature selection algorithm combines both new filter and wrapper.

In the next, details of the realization and designs for these methodologies will be presented in each chapter. Chapter 4 is the design of novel data preparing method for class imbalance problem. Chapter 5 is the design of novel filter and chapter 6 includes both design of wrapper and design of hybrid feature selection algorithms.

Chapter 4 - Design of Data Preparing Method

4.1 Introduction

As discussed in previous sections, data imbalance problem hinders the performance of machine learning. However, current solutions always add or delete instances that are used for training. This may cause problems in the later feature selection process, as irrelevant information may be added and useful information may be lost.

In this section, an Instance Selection based algorithm named Average Random Choosing Method (ARCM) will be designed to solve class imbalance problem in binary classification. Although it is not a basic part of feature selection, all data will be processed with ARCM to ensure a better performance of feature selection. Experiments and test results of ARCM are presented in section 7.3.

4.2 Motivation

The imbalance of data classes (where instances belonging to one class heavily outnumber instances in other classes) usually exist in credit datasets. The reason is that in real life the number of successful credit applications is usually larger than that of rejected applications. Thus, in the training of neural networks there should be more instances of approved applications in order to get a better scoring model. From the point of test data, as the original dataset is imbalanced, it is reasonable to keep the same ratio (i.e. approved/failed) in the test set. However, fairness cannot be guaranteed if the ratio between good and bad cases changes.

Another problem of data processing is the ratio of training-validation-test sets. All three sets should represent a relevant number of instances. Usually, more instances for training can lead to a higher chance of getting a better model. However, as the amount of data is limited, more data used for training means less for validation and testing. This will cause bad or unequal test performance. It is important to choose the best ratio since this affects the scoring model.

4.3 Average Random Choosing Method (ARCM)

To solve these problems, a method to process imbalance dataset is proposed in this research for binary classification problems. Suppose the total amount of instances is n, and the ratio of good applications in the dataset is p. Then the amount of good and bad applications is

Good applications: $p * n$,

Bad applicaions: $(1 - p) * n$

Then suppose the ratio of data used in training is t and in validation is v. Then it has

Training data: $n * t$

Validation data: $n * v$

Test data: $n * (1 - t - v)$

As the ratio of good to bad applications is hoped to stay the same in the training data (as in the original data), the training, validation and test data can be divided into good cases and bad cases. Table 1 shows the amount in each group:

Table 1 Amount of instances in each group

	Training dataset	Validation dataset	Test dataset
Good application	$n * p * t$	$n * p * v$	$n * p * (1 - t - v)$
Bad application	$n * (1 - p) * t$	$n * (1 - p) * v$	$n * (1 - p) * (1 - t - v)$

The flow of processing data is listed in Figure 11:

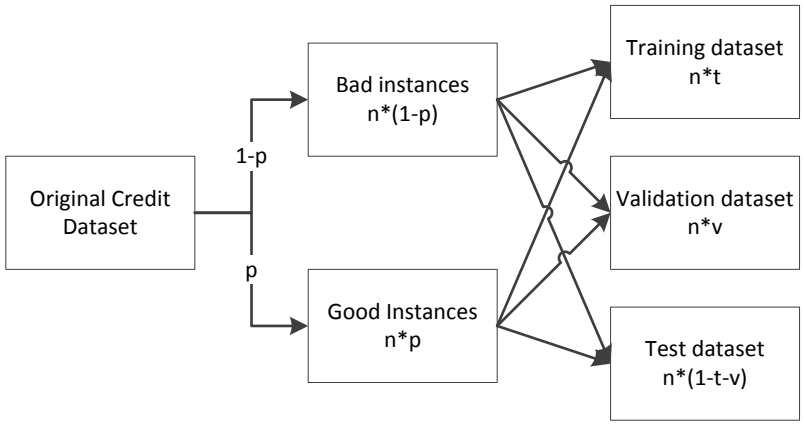

Figure 11 Flow of data processing and the amount of instances in each group

From the original dataset, two different kinds of data, bad instances and good instances are divided into two groups. Then both of them are divided into training, validation and test datasets randomly. This step should be repeated for each new network training session, which can minimize the effect of unordinary instances. This way, it is more likely to get a good data distribution which promotes a good scoring model.

4.4 Algorithm

Pseudo of this ARCM is described as follows:

Original Dataset D={xi, ti |i=0,1…n}

1. Select out all samples of class A A={xi, ti |ti=1},
2. Select out all samples of class B B={xi, ti |ti=2}
3. Select instances for test from class A TA={xi, ti |i=randperm(n*p*(1-t-v)), (xi, ti)∈A}
4. Select instances for test from class B TB={xi, ti | i=randperm(n*(1-p)*(1-t-v)), (xi, ti)∈B}
5. The test instances group T=TA∪TB
6. Samples in class A that are used to train and validate SA=A-TA
7. Samples in class B that are used to train and validate SB=B- TB
8. while(not reach the max epoch)
9. Choose out samples used for training in class A SAt={xi, ti |i=randperm(n*p*t), (xi, ti)∈SA}
10. Choose out samples used for training in class B SBt={xi, ti |i=randperm(n*(1-p)*t), (xi, ti)∈SB}
11. The training group is St= SAt+ SBt
12. The validation group is Sv=(SA- SAt)∪(SB- SBt)
13. Calculate output $y_i = \emptyset(\sum_{s=0}^{h} w_s \emptyset(\sum_{r=0}^{n} w_{sr} x_r))$, xr∈St
14. Update MLP by BP. Error signal is e(i)=ti-yi ti∈St
15. Validate MLP $y_i = \emptyset(\sum_{s=0}^{h} w_s \emptyset(\sum_{r=0}^{n} w_{sr} x_r))$, xr∈Sv
16. Performance of MLP $v = \frac{\sum_{i=0}^{n*v} e(i)}{n*v}$
17. if(v>v0) stop training
18. else v0=v
19. end while

By this method, all groups (training, validation, and test) have the same ratio of good to bad instances. Traditional 10-fold validation divides a dataset into 10 blocks of data randomly before training starts. The advantage is that the dataset can be trained 10 times with different datasets for training and validation. However, it is not really random choosing and is not suited for imbalanced datasets. In some extreme circumstances, there could be one group with only one class of data. This is obviously unable to judge the performance of the model. Our average random method chooses instances randomly from both classes of data. It can choose data randomly, which ensures more instance combinations can be used for training. Also, the training, validation, and test datasets have the same ratio of good to bad instances. This is specially designed for imbalanced

datasets by guaranteeing enough testing and training data from both classes. Comparison experiments will be presented below to choose the best training-validation-test ratio.

4.5 Summary

As a pre-processing step, ARCM enhances the performance of ANN that is used for binary classification problems from the point of instances used in training. Also it ensures an objective evaluation of learning algorithm performances, as it keeps the ratio between each class same in training, validation and testing. It does not add nor delete instances, which ensure a better performance of the later feature selection process.

ARCM will be used as the first step of MCFWFS algorithm, and the selected instances will be used in each training and evaluation process of this research. The experiments of ARCM will be demonstrated in section 7.4.

Chapter 5 - Design of Novel Filter

5.1 Introduction

In this section, a novel filter process based on the idea of “consistency” in feature selection will be designed. “Consistency” describes a relation between two instances. It is often defined by “inconsistency”: two instances are considered inconsistent if they match except for their class labels. When considering two attributes in dataset, if more instances are consistent on these two attributes, then these two attributes should have strong relationship, or sometimes called highly relevant. The new method, Consistency Concentration Based Feature Selection (CCBFS), is a feature ranking method fits for both binary classification problems and multi class recognition problems. It can rank features according to their relevance with the class label. It can handle with imbalance datasets and fits for both classification problems and multi class recognition problems. Another advantage of CCBFS is that it can handle with both continuous and nominal attributes naturally, without any discretise methods as used in traditional consistency based filters. Basic ideas of Consistency Based Feature Selection (CFS) have been introduced in section 2.5.1.1. The algorithm of CCBFS will be described along with a simple example. The experiments and test results of CCBFS comparing with other filters will be demonstrated in section 7.5.

5.2 Motivation of CCBFS

Consistency based feature subset selection has been proved to be competent regarding to improving machine learning performance and time. However, the shortage of it is also obvious. One problem is that consistency based measures can only be used on discrete features. Continuous features must be discretised before calculating the consistency rate, and the choice of discretization method may affect the production of feature subset. For example, when data include an attribute of ID numbers (such as social security number), this attribute is obviously consistent with the class label. So when the feature subset has this attribute, there will be no inconsistency in the data, but apparently it is irrelevant for rule induction (Dash and Liu, 1997).

Another thing is that in some circumstances, feature ranking is preferred, such as in some wrappers and when only a number of best features wanted to be selected. According to the literature review of this research, consistency based feature ranking method has never been defined, neither compared with other feature ranking methods.

Thus a novel consistency based filter that can provide feature ranking will be with high value.

5.3 Consistency Measure for Continuous Features

As what have been discussed, some continuous attributes can have extremely consistent value, but meaningless to machine learning. However, some continuous features may contain strong relation and at the same time with high consistency. Thus providing a new consistency measure for continuous features is with practical value.

In Liu's paper (Liu et al., 1998), the concept of consistency is defined by "inconsistency": two instances are considered inconsistent if they match except for their class labels.

Here the basic concept of this consistent measure is used. As the continuous features may have unreal large consistent value, this research defines the consistency of continuous feature in a slightly different way: define the consistency directly. Here is the new definition:

> **Def 1:** Consistency: Two instances are considered consistent if they match all values and class labels.

The only difference between this and the classical definition is the consistent number of "single matching pattern". For an attribute A_i, single matching pattern is the instance that has a unique attribute value. For example, when looking at attribute A1 in Table 2, instance 1 forms a single matching pattern because it match to no instance except itself. According to the classical consistency measure, the consistency number of this pattern is 1 (number of instance is 1, largest number of instances of class labels is 1, so the inconsistency number is 1 – 1 = 0). But in the novel definition of consistency, the consistency number is 0, because it matches to no other instance.

However, when counting the consistency number of nominal features, there is no difference between these two definitions. For example, A3 in Table 3, classical inconsistency number of A3 is (3-2)+(3-2)=2. So the consistency number is 6-2=4. In this case, the consistent number is 2+2=4, which is the same as Liu's result (Liu et al., 1998).

Table 2 Dataset for example

A1	A2	A3	A4	D
1	1	1	1	0
2	4	1	2	1
3	2	2	1	0
4	5	2	3	1
5	3	2	1	0
6	6	1	2	1

Table 3 Instance ranking results of the example dataset

A1	D
1	0
2	1
3	0
4	1
5	0
6	1

A2	D
1	0
2	0
3	0
4	1
5	1
6	1

A3	D
1	0
1	1
1	1
2	0
2	0
2	1

A4	D
1	0
1	0
1	0
2	1
2	1
3	1

5.4 Flexible Consistency Measure

While the change of consistency measure may decrease the consistency rate of continuous features, it still cannot truly reflect the relevance between one attribute and the target class. It can only focus on the matching patterns with large number of instances, but ignore those “single matching patterns”, which may also contain important information for machine learning.

Different from nominal features, the values of continuous features are with more information. For example, the “age” attribute in some medical diagnose dataset is considered as a continuous attribute. Larger value of this attribute may indicate high possibility of potential disease. So when evaluating continuous features, the variance of values should be taken into consideration.

To get the information from variance values, the definition of consistency is extended as follows:

Def 2: For attribute A_n and target class A_t, instances i and i+1 are considered consistent if

$$\nexists a | A_{n,i} \leq A_{n,a} \leq A_{n,i+1} \,\&\&\, A_{t,i} = A_{t,i+1} \tag{5.1}$$

where $A_{n,m}$ is the value of instance *m* at attribute *n*.

This flexible additional definition enlarges the consistency instances scope. Some single matching patterns are considered consistent with their "adjacent" instances. For example, A2 in Table 2 has a strict consistency number 0 when using the new definition mentioned before. By adding the flexible definition, the consistency measure becomes 6, as pattern 0 (including instances 1, 2 and3) has 3 consistency instances, and so does pattern 1 (instance 4, 5 and 6). At the same time, it will not affect the consistency number of A1, as instances 1 and 2 has different label, and instances 1 and 3 has one attribute between them. So the consistency number of A1 remains 0 as the strict definition, which is different from Liu's result.

5.5 Consistency Rate

By combining the flexible and strict definition of consistency, it has the following definition of consistency:

Def 3: Consistency: For a given attribute A_n and the class label A_t, two instances *i* and *j* are considered consistency if they match **Def 1** or **Def 2**. That is

$$A_{n,i} = A_{n,j} \&\& A_{t,i} = A_{t,j} \tag{5.2}$$

or

$$\nexists a | (A_{n,i} \leq A_{n,a} \leq A_{n,j} \,\&\&\, A_{t,i} = A_{t,j}) \tag{5.3}$$

The Consistency Rate is the sum of all consistency counts divided by the total number of instances. That is

$$Consistency\ Rate = \frac{sum\ of\ consistency\ count}{number\ of\ instances} \tag{5.4}$$

As the sum of consistency count is no smaller than 0 and no bigger than the total number of instances, so it has

$$0 \leq consistency\ rate \leq 1 \tag{5.5}$$

The consistency rate equals to 0 when an attribute has different class labels between any adjacent instances (A1 in Table 2). It equals to 1 when an attribute has strict linear correlation with the target class (A2 in Table 2).

5.6 Concentration Measure

Filters take no consideration of the machine learning algorithm, but the aim of filters is always increasing the prediction accuracy. Take classification problem for an example. Many learning algorithms try to divide input space into separate regions by linear boundaries, such as support vector machine (SVM) and multilayer perceptron (MLP). From the point of simplifying learning algorithms, it would be beneficial to find an intuitive meaning of continuous feature selection. The “good” features should have a more concentrate value distribution: the input value of instances with the same class label should concentrate in a small range.

Take blood pressure in medical diagnose dataset for an example. For one given disease, if the blood pressure of potential patients concentrates on some areas, then this attribute may contain useful information when predicting. A more direct example can be seen in Table 4.

Table 4 Another example

A1	1	2	3	4	5	6	7	8	9	10
A2	1	2	3	6	7	8	4	5	9	10
T	0	1	1	0	0	0	1	1	0	0

According to the new consistency measure, the number of consistency of A1 and A2 are counted as follows:

A1 = |{2,3}|+|{4,5,6}|+|{7,8}|+|{9,10}| = 9

A2 = |{2,3,4,5}|+|{6,7,8,9,10}| = 9

However, the distribution of A1 values is more like a random distribution, while A2 has clearly organized distribution: instances with larger values are more likely to have class label 0. From the point of manual classification, A2 is more welcomed and trusted than A1, because its values tend to "concentrate" on specific areas for each class labels.

To make the new filter more biased towards attributes with high concentration, this algorithm adds "concentration offset" into the consistency rate by function $\gamma(A_i)$ where A_i is a given attribute. A consistent pattern is a group of instances that consistent with each other.

The concentration offset of an attribute is defined as follows:

> **Def 4**: Concentration: an attribute is concentrate when a value based division of instances can ensure each instance group forms an unique consistent pattern, and the number of group is equal to the number of class labels.

The consistency offset is defined by

$$\gamma(A_i) = \frac{kp}{n} \tag{5.6}$$

where p is the number of consistent patterns in attribute A_i, and n is the total number of instances. Here k is an empirical predefined value and is always between -1 and 0. As p is always positive and smaller than the number of instances, so it has $k < \gamma(A_i) \leq 0$.

5.7 Consistency Concentration Rate (CCR)

By adding concentration rate, the consistency concentration rate (CCR) is:

$$CCR(A_i) = \frac{c}{n} + \frac{kp}{n} = \frac{c + kp}{n} \tag{5.7}$$

where *c* equals to the consistency count of attribute A_i.

Prove 1: if $-1 \leq k < 0$ then $0 \leq CCR(A_i) < 1$

When *c*=0, which means there is no consistency pattern in that attribute, then *p*=0. So *CRR*=0.

When $c \neq n$, which means the attribute includes some consistent patterns, then *p*>0. That's because in a classification problem there must be more than 2 classes. So there is $0<p<n$ and $-n<kp<0$.

When counting the consistent instances, there must be at least 2 instances that form a consistent pattern. So the number of consistent patterns must be smaller than the number of consistent instances. That means $c>p$ and $c>-kp$. So it has $c+kp>0$ and $CCR = \frac{c+kp}{n} > 0$.

Also it is known that $kp<0$, so $c + kp < c \leq n$ and $CCR = \frac{c+kp}{n} < 1$.

Thus, it has $0 \leq CCR(A_i) < 1$ when $-1 \leq k < 0$.

From this prove it can be seen than CCR cannot be equal to 1. To make the CCR be exactly in the closed interval 0-1, there is a slight modification of the CCR calculation.

$$CCR(A_i) = \frac{c + kp}{n - l} \tag{5.8}$$

where *l* is the number of classes.

Prove 2: if $-1 \leq k < 0$ then $0 \leq CCR(A_i) \leq 1$

As it has been proven that $\frac{c+kp}{n} \geq 0$, so $CCR = \frac{c+kp}{n-l} \geq \frac{c+kp}{n} \geq 0$ as *l* is positive.

If $-1 \leq k < 0$ then $c + kp \leq c - p$. To maximize *c-p*, *c* should be equal to *n* which means all instances are consistent in the given attribute. In this situation, the minimal value of *t* is equal to the number of classes. So it has $c - p \leq n - l$.

Thus it has $CCR = \frac{c+kp}{n-l} \leq 1$. *CCR*=1 when *k*=-1, *c*=*n* and *p*=l.

5.8 CCBFS Algorithm

The calculation of CCR is different from Liu's consistency measure, as this research takes continuous features as themselves but in Liu's algorithm they are discretised first and treated as nominal features. So hash table cannot be used to simplify CCBFS.

The procedure of CCBFS algorithm is as follows.

```
CCBFS Algorithm
Input: dataset D(A, T), number of instances n
For each attribute Ai
        Find all consistent patterns in Ai
        Count the number of patterns
        Calculate CCR by equation (5.13)
Rank features according to their CCRs from high to low
Select out N best features in the rank list
```

The most important step is to find all consistent patterns in Ai. In the program, it counts the number of consistent instances and calculates *CCR* directly to simplify the algorithm. Here is the algorithm of finding all consistent patterns in CCBFS

```
Find all consistent patterns for Ai
Initialize consistent pattern P as null
While D is not null
Find smallest value Ai,j in data table D(Ai, T) and get the instance I(Ai,j, t)
If P is null or Ai,j equals to all instances in data table P at attribute Ai
        P=P+I
Else
        If t equals to all instances in P at T
                P=P+I;
        Else
                For all instances in P that are not equal to Ai,j at attribute Ai
                Count the number of instances in this group Pn as pnt
                CCN = CCN + pnt – k;
                P=P-Pn+I;
```

In average cases, the efficiency of this process in $O(n^2)$.

5.9 Summary

In this chapter the novel filter algorithm CCBFS is described in detail. This consistency based novel filter can rank features according to their relevance to the class label. The CCR is designed according to the relation between target class and each candidate attribute. Features with higher relevance to the targe class are ranked higher.

One improvement comparing with traditional consistency based filters is that the novel filter is specially designed to handle both nominal and numeric features, which avoid using discretised methods. This is achieved by a novel definition of consistency.

The novel filter can be utilized on many different categories of dataset, such as imbalance dataset, classification dataset and multi class recognition dataset. Experiments of CCBFS are in section 7.5. CCBFS will be proved to be the best choice for the later designed hybrid feature selection process in chapter 7.

Chapter 6 - Designs of Wrapper and Hybrid Feature Selection Algorithm

6.1 Introduction

After the design of filters (CCBFS), a novel wrapper and a novel hybrid feature selection are designed in this section. The novel wrapper named Accumulate Elitism Genetic Algorithm (AEGA) is based on GA and using an elitist list to accelerate convergence. It specifies ANN as the machine learning algorithm and a special design of storing best findings is used to decrease ANN training times. AEGA solves the long training time problem of ANN wrappers.

A novel hybrid feature selection algorithm, MCFWFS, is designed by combining CCBFS and AEGA in multiple ways. It combines CCBFS into the initialization process of AEGA, and utilizes the feature ranking results of CCBFS to build up local search operators for hybrid GA. MCFWFS can achieve the main aim of this research: improving the performance of ANN from the aspect of both computing speed and prediction accuracy.

Experiments and results of MCFWFS are demonstrated in section 7.6.

6.2 Accumulate Elitist Genetic Algorithm for Feature Selection

In this section a new GA for feature selection is proposed. As it is designed for a hybrid process of feature selection, experiments will only be designed for the hybrid process.

6.2.1 Motivation

A critical problem in wrapper for MLP is that the performance of MLP is unstable. In the training of MLP, weights of each neuron are typically randomly initialized. So cross validation is essential for a reliable evaluation of MLP. Also back propagation (BP) in training takes longer time than other machine learning algorithms such as Naïve Bayes and Decision Trees. So for MLP wrapper, the number of calling fitness function should be reduced as much as possible. In the meantime, searching solutions with highest prediction accuracies needs a large searching scope. The dilemma of searching more areas and fewer callings of fitness functions will be solved by this research.

Some parts of Simple GA (SGA) wrapper are modified and the novel wrapper AEGA will be introduced in detail in this section..

6.2.2 Initialization

All GA methods start from initializations of the first generations. As what have been mentioned, binary bits strings can represent the candidate feature group. With a random function *random_int(n)* which can generate a random integer number between 0 (include) and n (exclude), the initialization algorithm of population P is as follows:

```
Initialization:
for(i from 1 to |P|)
        number_selected = random_int(|p|)
        for(j from 1 to number_selected)
                set gene random_int(|P|) in ith chromosome as 1
```

It should be noticed that this algorithm is just for SGA and will be modified later in the new hybrid GA wrapper. But for now it can be used as a standard initialization process.

6.2.3 Fitness Function

In this research a new penalty value in fitness function is designed for a smooth reduction of features. In SGA for feature selection the penalties are designed to induce GA converging to solutions that with a desired number of features. For example, in paper (Oh et al., 2004) it is defined as $Pe(A_c) = w(|A_c| - d)$, which means those solutions that have more or less features than d will be punished. This is reasonable when people have some basic ideas about feature reduction, and many researches have proved that penalties with a predefined number of features perform well.

However, to those datasets with no information, this desired number of features is difficult to define. Sometimes several different values are tried and choose out the one with best machine learning performance. This could be tough considering that parameter setting of GA is already a tough problem. To simplify SGA settings, an adaptive penalty in this research is defined as:

$$Pe(A_c) = w\left(\frac{|c|}{|P|}\right)^2 \tag{6.1}$$

where w is a fixed value 0.01, $|c|$ is the number of features in candidate feature set and $|P|$ is the number of features in original dataset. This penalty increases fast when applied on large feature subsets, which gives a high pressure to the direction of reducing features. On the other side, the penalty decreases tiny in the low number area, which almost has no influence to the fitness function.

With this penalty, the fitness function is defined as:

$$fitness(c) = P(A_c) - Pe(A_c) = AUC(A_c) - w\left(\frac{|c|}{|P|}\right)^2, w = 0.01 \tag{6.2}$$

The calculation of AUC is introduced in section 7.3.

6.2.4 Scale Fitness

To avoid fast converge into local optimum in the beginning and distinguish better solutions when converged, this research involves a linear scale process of the fitness scores. The linear scale function is:

$$f' = a + bf \tag{6.3}$$

where f' is the scaled fitness and f is the raw fitness. To maintain a certain relationship between the maximum fitness individual in the population and the average population fitness, a and b can be calculated by the fact:

$$\begin{cases} f'_{max} = f_{ave} * C \\ f'_{ave} = f_{ave} \end{cases} \tag{6.4}$$

where C is a scaling constant that specifies the expected number of copies of the best chromosome in the next generation (usually it has C=2). Another restriction is that all values of f' should be positive. As f' is monotonic increasing, to make it positive only needs the following restriction:

$$f'_{min} = af_{min} + b > 0 \tag{6.5}$$

Combining these three equations above, the value of a and b can be calculated by

$$\begin{cases} Cf_{ave} = af_{max} + \mathrm{b} \\ f_{ave} = af_{ave} + \mathrm{b} \end{cases} \text{when } f_{min} > -\frac{b}{a}$$

(6.6)

So it has

$$\begin{cases} a = \dfrac{(C-1)f_{ave}}{f_{max} - f_{ave}} \\ b = \dfrac{f_{ave}(f_{max} - Cf_{ave})}{f_{max} - f_{ave}} \end{cases} when\ f_{min} > \frac{Cf_{ave} - f_{max}}{C-1}$$

(6.7)

For the case of $f'_{min} < 0$, which is $f_{min} < \frac{Cf_{ave} - f_{max}}{C-1}$, the maximum value should not be reached. So the following equations should be satisfied:

$$\begin{cases} f'_{ave} = f_{ave} \\ f'_{min} = 0 \end{cases}$$

(6.8)

which is

$$\begin{cases} f_{ave} = af_{ave} + \mathrm{b} \\ 0 = af_{min} + b \end{cases} when\ f_{min} < \frac{Cf_{ave} - f_{max}}{C-1}$$

(6.9)

The solution is

$$\begin{cases} a = \dfrac{f_{ave}}{f_{ave} - f_{min}} \\ b = \dfrac{-f_{min}\, f_{ave}}{f_{ave} - f_{min}} \end{cases} when\ f_{min} < \frac{Cf_{ave} - f_{max}}{C-1}$$

(6.10)

So the scaling algorithm is:

Scaling fitness algorithm:

Find the maximum f_{max} and minimum f_{min}

Calculate the average fitness in population f_{ave}

If $f_{min} > \frac{Cf_{ave}-f_{max}}{C-1}$

Set $a = \frac{(C-1)f_{ave}}{f_{max}-f_{ave}}$, $b = \frac{f_{ave}(f_{max}-Cf_{ave})}{f_{max}-f_{ave}}$

Else

Set $a = \frac{f_{ave}}{f_{ave}-f_{min}}$, $b = \frac{-f_{min}f_{ave}}{f_{ave}-f_{min}}$

Calculate f' by $f' = a + bf$

The linear scaling is simple and proved to be effective for feature selection. In this algorithm it will be combined with other chromosome selecting methods to achieve better performance.

6.2.5 Accumulate Elitist Selection, Crossover and Mutation

This algorithm implements an Accumulate Elitist Selection (AES) aiming at fast converging speed and robust outcomes. An elite list is used to store the best chromosome in each generation. These chromosomes will be revaluated again in the next generation to avoid abnormal prediction accuracies. Then its fitness will be updated to the average value of all tests. Then this average fitness will be compared with others in the select process in the next generation. The algorithm of AES is described as follows:

Algorithm of AES

For population P_n in Generation n:

Find the best chromosome c_{best} in P

For all chromosomes c[i] in elite list

$c[i]_{ave\_fitness}$ =($c[i]_{ave\_fitness}$ *$c[i]_{best\_count}$+$c[i]_{fitness}$)/($c[i]_{count}$+1)

$c[i]_{count}$ = $c[i]_{count}$+1

if(c_{best} = c[i])

$c[i]_{best\_count}$=$c[i]_{best\_count}$+1

if c_{best} not in elite list

push c_{best} -> elist_list and stored as c[m+1]

$c[m+1]_{fitness}$ = $c_{best\ fitness}$

$c[m+1]_{best\_count}$ = 1;

generate new population P_{n+1}

For each c[i] in elite list

copy c[i] into P_{n+1}

put localsearch(c[1]) into P_{n+1}
For each 2 of the rest chromosomes in P_{n+1}
select 2 chromosomes from P_n by scaled roulette wheel selection
crossover
mutate

In this algorithm, $c[i]_{best\_count}$ is used to record the number of times one chromosome in elite list is evaluated as the best in population. $c[i]_{count}$ is the number of evaluation times of *c[i]*. This is for the calculation of average fitness, and later for checking the stop criteria. *Localsearch()* function can generate "nearby" chromosomes to achieve faster local search. This will be described in the hybrid feature selection algorithm later (section 6.5.2). For now it can be seen as a way of mutation.

AES keeps the best chromosomes in the next generation. As the training of MLP is a random process, unstable fitness values can be avoided by calculating the average accuracy in all generations. Each chromosome in elite list represents a local optimum, and the local search method will make it easy to search around that location. It should be notice that AES should be used in combination with hybrid GA and the stop criteria that will be described later.

The accumulated elite group also ensures that better features are more likely to be inherited into more chromosomes of the next generation. This means the searching pressure is higher in local optimum area and quickly converges to the optimum.

The algorithm of roulette wheel selection is as follows:

Roulette wheel selection: return the selected chromosome
Calculate the sum of fitness as F_{sum}
Generate a random number f_{random} = random_float*F_{sum}
For(i from 0 to |P|)
$f_{random} = f_{random} - c[i]_{fitness}$
if (f_{random}<0)
retrun c[i]

where the random_float function can generate a float number within [0, 1).

With a predefined crossover rate $p_{crossover}$ and mutation rate p_{mutate} the algorithm of these two operators are as follows:

```
Crossover for two selected chromosomes c1 c2:
if (random_float < pcrossover)
        crosspoint = random_int(numAtt-1)
        for all bits in c1 and c2 after the crosspoint
                swap(c1, c2)
return c1, c2
```

```
Mutate for a chromosome c
for each bit in c
        if (random_float < pmutate)
                flip the bit
return c
```

6.2.6 Stop Criterion

To achieve a short searching process, there are three stop criteria in this algorithm.

(1) when the number of generations exceed maximum

(2) when the length of elite list equals to the half size of population, which means no new best chromosome can be generated

(3) when a chromosome in elite list has been evaluated as the best in generation for n times (n>5)

The first criterion is the same as SGA's. In many research, this maximum generation is set very high, and usually relevant to the number of attributes. In this research this number is set to be large enough that should hardly be reached.

Criterion 2 is met when there are too many features in elite list that no new chromosome will emerge in the later generations. This criterion can be controlled by the size of population. So in AEGA, the population size is preferred to be larger than SGA. For example, in SGA a typical size of population is 30. Here it should be set according to the number of attributes in original dataset. For small dataset ($numAtt<20$) 30 can be enough, but for medium ($20<numAtt<50$) 50 is a better choice. For those large datasets, this number could be between 100 and 200. The reason for a large population size is that since it applies harsh stop criteria and accumulates elite chromosomes, more

chromosomes in population means larger search scope in one generation. Another reason is for longer elite list, which can absorb more potential best solutions.

A good search of the new AEGA is more likely to stop by criterion 3. If a chromosome is evaluated as the best in population for many times, then naturally it can be seen as a local optimum and a robust solution.

To be fair for each chromosome in elite list, when the algorithm stops all of them should be evaluated at least 10 times. So those chromosomes that pushed into elite list very late must supply some more evaluations and update their average fitness. Then the solution with the highest fitness returns as the searching result of AEGA.

6.2.7 Overall Algorithm

The algorithm of AEGA is as follows:

```
AEGA()
Initialize population
repeat{
        calculate fitness of each chromosome
        scale fitness
        do accumulate elite selection, including crossover and mutation
        replace P with new chromosome sets
}until (stop criteria)
```

Parameter setting:

Population size (pop_size):
- depend on dataset: Small: pop_size =30
  Medium: pop_size =50
  Large: pop_size =100

Maximum generation (max_generation): 100 or larger

Crossover rate ($p_{crossover}$): 0.6

Mutation rate (p_{mutate}): 0.033

Scale pressure (C): 2

Fitness penalty weight (w): 0.01

6.3 Hybrid Genetic Algorithm for Feature Selection

To solve the current problems of wrappers for ANN, such as long computing time and low accuracy, this research proposes a hybrid feature selection algorithm called Multi Combined Filter-Wrapper Feature Selection (MCFWFS). This new algorithm is based on Hybrid Genetic Algorithm (HGA). Both CCBFS and AEGA are utilized in MCFWFS to achieve fast computing speed and higher prediction accuracy.

As mentioned in literature review, applying HGA on feature selection starts from the research in paper (Oh et al., 2004). The basic idea of HGA in this research is from works in paper (Zhu et al., 2007), but with a throughout improvement. A detailed explanation of HGA in (Zhu et al., 2007) has been given in section 2.7.1.

To cover the limitations of HGA, this research proposed a novel hybrid feature selection algorithm MCFWFS. This new algorithm combines CCBFS with AEGA from two aspects: 1. using CCBFS to initialize the first population of AEGA. And 2. accelerate AEGA by LSO which are designed according to the feature ranking list provided by CCBFS. Both the pre-selection process and the LSO will be introduced in the next sections.

6.4 Filters as Pre-Selection for Wrappers

All searching methods in wrappers, especially GA based searching strategies, starts with one or a set of feature subsets. In the newly designed AEGA, as it tends to have a quick converge around the best chromosome in the first generation, a high quality initialization is essential. Another aim of utilizing filters before wrappers is controlling the number of features. A random initialization may generate a chromosome with lots of features and relatively high fitness score. The local search around this chromosome is not as expected as on chromosomes with proper size. A filter-controlled initialization can create some chromosomes with *n* best features, which makes these chromosomes highly likely to be passed into the next generation.

In the initialization of AEGA, m chromosomes are initialized based on ranking lists of filters. There are two searching strategies for creating these m chromosomes: (1) sequential forward search (SFS) (2) dataset adapted select (DAS). In dataset U = F*I where F is the set of all features f and |F| is the number of features. If X denotes the

subset of selected features and Y denotes the unselected, then algorithms of SFS and DAS are:

SFS algorithm:

Set $X = \emptyset$, $Y = U$

While $Y \neq \emptyset$

Find the highest ranked feature f in Y

$X = X + f$, $Y = Y - f$;

Calculate the accuracy of the specified classifier on X

Return the best *m X*s and encode as binary strings

DAS algorithm:

$s = \sqrt{|F|}$

Select top s features in the ranking list

Select top 2s features in the ranking list

For each of the other m-2 subsets

rand = random_int(s, 2s)

Select top *rand* features in the ranking list

Return all subsets and encode as binary strings

Obviously, SFS can generate feature subsets with higher classification accuracies; while at the same time takes longer time on evaluation. In this research, as BP is a kind of time costing training algorithm, DAS algorithm is preferred. To ensure a larger searching scope, m = 2 is a good choice and will be used in experiments.

Thus, the initialization procedure of MCFWFS is as follows:

Initialization algorithm in MCFWFS

Apply filter algorithm (CCBFS for example) on original dataset U

Generate feature ranking list

rl = {f1, f2, … , fn} where n = |F|

Chromosome {c1, c2} = DAS(m=2)

For(i from 3 to population size)

Ci = Simple GA initialization from dataset U

This filter based initialization method with DAS strategy can give the first generation two chromosomes with proper sizes. As CCBFS has been proved to be competitive and

sometimes even better than other filters, these two initial chromosomes are likely to have high fitness scores. Comparing with random initialization process, this filter-wrapper combined algorithm can accelerate the GA based searching by providing two good solutions in the very beginning. With other chromosomes initialized in a random way, this initialization method can also maintain a large searching scope.

6.5 Filters for LSO in wrappers

6.5.1 Design of LSO

In AEGA, the elitist features are altered by LSOs to perform fast converging to local optimum. Here the method of LSO with feature ranking list will be presented.

For dataset U = F * I, X denotes selected feature set and Y denotes unselected. First, operations of “add” and “del” are defined as:

add: if ($X \neq U$), choose one feature f from Y and move f from Y to X

del: if($X \neq \emptyset$), remove one feature f from X and move f into Y

The basic idea of add and delete operations are from the HGA in research (Oh et al., 2004). The feature choosing strategies of previous researches focus on adding the best features and removing the worst features. This strategy is reasonable when each of the adding and deleting combinations is evaluated and the best combination is selected. However, the evaluation process is time consuming, especially for MLP training process. A much simple way of adding and deleting operations is using Roulette Wheel selection described in section 2.8.3.1. Here the random selection strategy uses CCR of each feature generated by CCBFS. The CCBFS process has been applied before AEGA (that is, in the last section). All CCRs are scaled by linear scale function and it is as same as in section 6.2.4. An example of add/del operation is displayed in Figure 12.

Figure 12 only shows a possible selection of Roulette Wheel. In del operation, as features with higher CCR are expected to be reserved and those with lower CCR are expected to be removed, the size of each “slice” that represent the corresponding feature is (1 – CCR) (it has been proved in section 5.7 that $0 \leq CCR(A_i) < 1$). As CCBFS takes no consideration of the correlation between candidate features, some of them with low CCRs may increase machine learning performance in conjunction with others. In

this situation, while those "good" features have higher opportunities to be selected, there is still chance for those "bad" features.

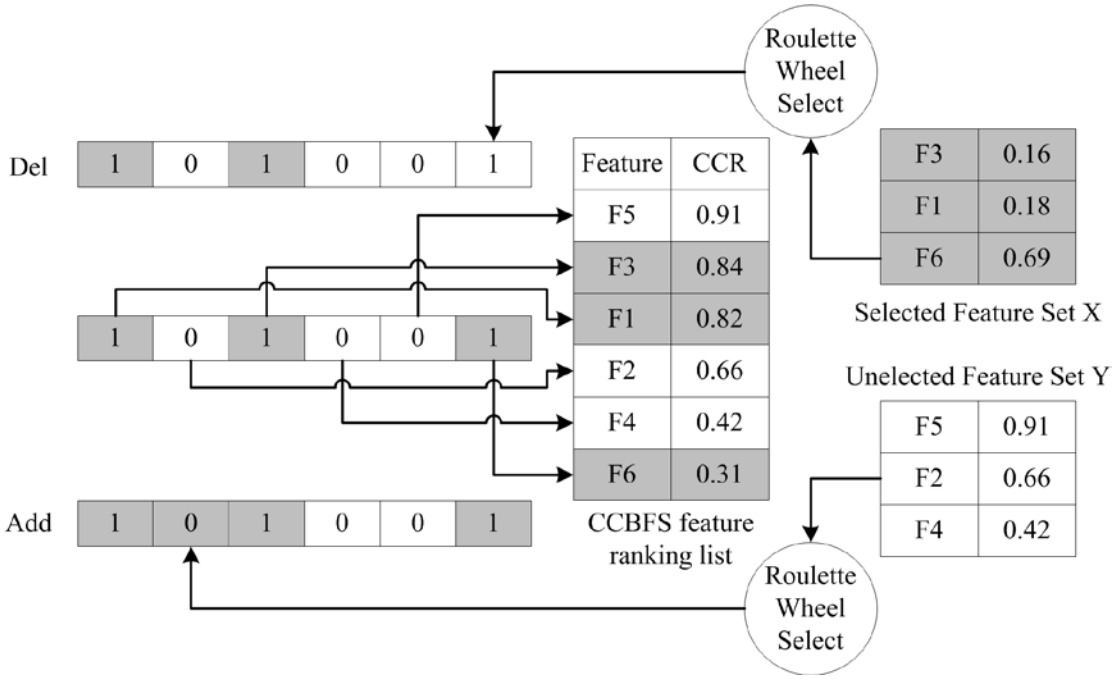

Figure 12 LS operations: add and del

6.5.2 Local Search Strategy

As both add and del operations cannot guarantee generating a better chromosome, the local search strategy is vital for finding the local optimum quickly. One recessive demand of searching is to find the chromosome with high fitness and as few features as possible. So searching around smaller feature sets are more desired than to larger sets. Another dilemma of local searching is the intensity, that is, how many features should be added or removed to one chromosome. This research designs a dynamic local search strategy to achieve these two aims.

The algorithm of local search is as follows:

Localsearch() algorithm
For chromosome c in elitist list
    c_add = c, c_del = c, c_ls = c
    $l = 2 + \log_2 c_{count}$
    apply "add" *//2* times on c_add
    apply "del" /times on c_del
    c_add = c_add *XOR* c

```
        c_del = c_del XOR c
        c_ls = c OR c_add
        c_ls = c_ls ANDNOT c_del
return c_ls
```

where c_{count} denotes the number of evaluations on c. XOR, OR and ANDNOT are three logical operators for binary bits strings.

In each generation, the intensity of local search for each chromosome in elitist list depends on the times that it is evaluated. The longer a chromosome stays in elitist list, the larger the searching scope around it will be. This is a stimulation of human searching process: to find a local optimum around a "good" point, we always starts from the nearby fields, and then extend to far spaces. This dynamic local search strategy works in the same way: starts searching by adding 1 and removing 2 features, then gradually add and delete more features. The value of *l* will not grow too large, for example, within 64 generations *l* will be less than 8. This avoids excessive searching at the end of AEGA, while still keeps raising the searching intensity.

Another noticeable point is that the added features are always fewer than those removed. The motivation is that searching subsets with fewer features is more worthwhile than spending time on large subsets. High quality feature subsets with fewer features are less likely to be missed under high searching intensity on them.

To avoid conflicts between added and removed features (that is, remove an added feature or add a removed feature in the same generation), all LSOs are applied directly on the original chromosome. Then all added and removed features are selected out and marked by c_add and c_del after XOR operation. In the last step these features are added or removed by OR and ANDNOT operations to avoid conflicts.

6.6 Overall algorithm of MCFWFS

After all, the algorithm of Multi-Combined Filter Wrapper Feature Selection is shown in Figure 13.

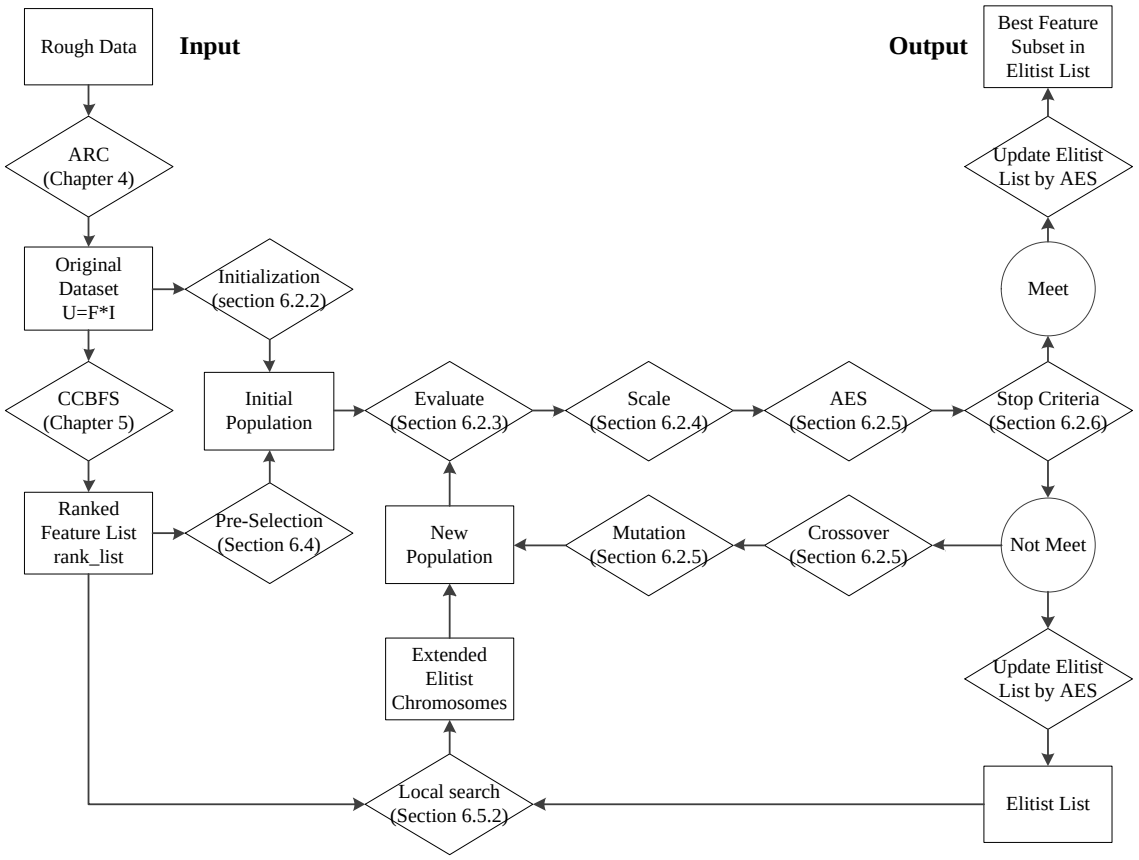

Figure 13 Overall algorithm of MCFWFS

Details of each step are in the corresponding chapters and sections.

6.7 Summary

In this chapter, a novel wrapper for ANN is designed. AEGA solves the long computing problem of traditional ANN wrappers by using elitism list in GA, and a new way of recording best findings in GA to minimize ANN training times. Several other operators and steps in GA are also modified to adapt to hybrid feature selection.

The new hybrid feature selection algorithm MCFWFS combines CCBFS into AEGA in two ways: apply CCBFS before AEGA as a pre selection, which initializes two chromosomes in the first generation; and use CCBFS in the LSO of AEGA to accelerate converging process. After all, the overall structure of MCFWFS, including the pre-processing step ARCM, is described in figures. Experiments of MCFWFS will be introduced in the next chapter.

Chapter 7 - Experiments and Results

7.1 Introduction

In this section the experiments of this thesis will be demonstrated.

First, all datasets used in this research will be introduced in section 7.2, including 15 different datasets. These datasets cover many real world problem fields, including classification and multi class recognition; with different kind of attributes such as nominal, continuous and hybrid; with numbers of attributes from 4 to 7071. These datasets are carefully chosen to test the performance of new algorithms.

Then the experiments of each novel algorithm are shown according the designing sequence in this thesis. The experiment of novel data preparing method ARCM is presented first in section 7.4. Then the experiment settings and results for the new filter CCBFS is shown in section 7.5. To better improve the performance of ANN, MCFWFS is designed and the experiment results are shown in section 7.6.

7.2 Datasets

In this section, datasets used in this thesis will be introduced. All dataset used in this experiment are from UCI (Lichman, 2013) machine learning repository and Feature Selection Dataset (FSD) (Li et al., 2016).

According to the categorization of problem sizes described in (Oh et al., 2004) and (Kudo and Sklansky, 2000), dataset with less than 20 features are considered small dataset, and those with more than 50 can be seen as large dataset. Others are medium dataset whose number of attributes is between 20 and 50. In this experiment, dataset of all categories will be used. To be more specific, it will use 2 small, 4 medium and 2 large dataset.

As the above categorization is proposed 15 years ago, this research supplies the idea of “extremely large dataset” which contains more than 100 attributes. 4 extremely large dataset will be used to test the performance of feature selection algorithms.

1. Germen Credit (Statlog)

This dataset contains 20 features and one binary label attribute. This dataset classifies people described by a set of attributes as good or bad credit risks. In these 20 attributes there are 3 continuous and 17 nominal ones. In its 1000 instances, 700 of them are

classed as “good” and 300 as “bad”. So this is slightly imbalance dataset with number of nominal attributes.

2. Australia Credit (Aus)

This dataset concerns credit card applications. There are 6 numerical and 8 categorical attributes. There is a good mix of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

3. Waveform (Wf)

This dataset contains 3 classes of waves and 40 attributes. The latter 19 attributes are all noise attributes with mean 0 and variance 1. It has 5000 instances, and in some tests that only consider binary classification, instances with class 1 and 2 form the binary waveform dataset.

4. Ionosphere (Io)

This dataset includes classification of radar returns from the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. All 34 attributes are continuous. 225 of the total 351 instances are in class “g” so it can be seen as imbalanced dataset.

5. QSAR biodegradation (Bio)

The data have been used to develop QSAR (Quantitative Structure Activity Relationships) models for the study of the relationships between chemical structure and biodegradation of molecules. This dataset contains values for 41 attributes (molecular descriptors) used to classify 1055 chemicals into 2 classes (ready and not ready biodegradable). One third of the total instances belong to “RB” class and others belong to “NRB”.

6. Sonar (So)

The task of Sonar is to discriminate between sonar signals bounced off a metal cylinder and those bounced of a roughly cylindrical rock. It contains 208 instances and 60 attributes where all values in the range 0.0 to 1.0. Each number represents the energy

within a particular frequency band, integrated over a certain period of time. The class labels contain “R” (rock) and “M” (metal cylinder). It contains 111 patterns with “R” and 97 with “M”.

7. Lung cancer (Lc)

The lung cancer dataset described 3 types of pathological lung cancers. It contains 32 instances and 56 attributes that are all nominal. The class label contains 3 values from 1 to 3, with 9, 13 and 10 instances respectively. All attribute values are nominal.

8. Vehicle (Ve)

The task of Vehicle dataset is to recognize a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The original purpose was to find a method of distinguishing 3D objects within a 2D image by application of an ensemble of shape feature extractors to the 2D silhouettes of the objects. The dataset contains 18 attributes and 946 instances. The four types of vehicle are: OPEL, SAAB, BUS and VAN. All attribute values are continuous.

9. Lung discrete (Ld)

This is one of the four extremely large dataset got from Feature Selection Datasets (FSD) (Li et al., 2016). It contains 326 attributes and 1 target class that contains 3 labels. All instances are discrete.

10. Colon (Co)

Colon is another extreme large dataset from FSD (Li et al., 2016). It contains 2000 attributes and 62 instances, with 2 different class labels. It is a nominal dataset and each attribute has 3 possible values. The percentage of two labels is 40:22.

11. Lymphoma (Ly)

Lymphoma is another extreme large dataset from FSD (Li et al., 2016). It contains 4026 candidate attributes and one class attribute. There are 96 instances with 9 class labels. It is a typical multi class recognition problem with very high dimensions.

12. Leukemia (Le)

Leukemia dataset is another extreme large dataset from FSD (Li et al., 2016). It contains 7071 attributes and 72 instances with 2 class labels (47 labels with "-1" and 25 with "1"). All attribute values are nominal.

13. Pima Indians Diabetes (Pima)

Pima dataset is from National Institute of Diabetes and Digestive and Kidney Diseases. The goal is to forecast the onset of diabetes mellitus. There are 8 numeric features and one target class (0 and 1). It contains 768 instances. This dataset will be used in HONN.

14. Blood Transfusion Service Center (Blood)

The Blood dataset was taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan. There are 4 numeric features and one binary feature of class labels. This dataset will be used in HONN.

15. Liver Disorders (Liver)

There are 6 numeric features in this dataset and 345 instances. These features are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. It is a binary classification problem and will be used in HONN.

7.3 Area Under Curve

Traditionally, classification accuracy is used to judge the "goodness" of feature group. Recently the Area Under Curve (AUC) has been proved to be a good alternative (Fawcett, 2006). It was firstly used as evaluation function in feature selection to rank features in (Chen and Wasikowski, 2008) and (Wang and Tang, 2009). It was extended to multi-class in (Wang and Tang, 2012) and test results showed that AUC has better performance than traditional overall accuracy (OA).

Define an experiment from P positive instances and N negative instances for some condition. The four outcomes can be formulated in a 2*2 contingency table or confusion matrix, as in Table 5 (Fogarty et al., 2005):

Table 5 Four outcomes of an experiment

		Condition		
		Positive	Negative	
Test outcome	Positive	**True positive**	**False positive (Type 1 error)**	Precision= $\frac{\sum Ture\ positve}{\sum Test\ outcome\ positive}$
	Negative	**False negative (Type 2 error)**	**True negative**	Negative predictive value= $\frac{\sum True\ negative}{\sum Test\ outcome\ negative}$
		Sensitivity= $\frac{\sum True\ positive}{\sum Condition\ positive}$	Specificity= $\frac{\sum Ture\ negative}{\sum Condition\ negative}$	Accuracy

A receiver operating characteristic (ROC) space is defined by false positive rate (FPR) and true positive rate (TPR) as x and y axes respectively. Write the probability for belonging in the class as a function of a decision/threshold parameter T as $P_1(T)$ and the probability of not belonging to the class as $P_0(T)$. The false positive rate FPR is given by FPR(T) $= \int_T^\infty P_0(T)\, dT$ and the true positive rate is TPR(T) $= \int_T^\infty P_1(T)\, dT$. The ROC curve plots parametrically TPR(T) versus FPR(T) with T as the varying parameter. The area under the curve (AUC) of ROC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative') (Fawcett, 2006).

As AUC is contains information of precision, it can be seen as an alternative to traditional classification accuracies. It is even better, because it is not affected by imbalanced class. So AUC is a more objective choice when evaluating ANNs.

For this reason, AUC will be used as the evaluation of machine learning algorithms for all experiments to replace classification accuracy.

7.4 Experiments and Results of Novel Data Preparing Method ARCM

This section designs experiments to determine a competitive MLP model for credit scoring. The first aspect is to find the best amount of data used in training, validation, and testing, respectively. Then based on the results, the second aspect is to focus on

ARCM. Thirdly, discuss the number of hidden neurons by training each model 20 times with different initial weights for each kind of model in all experiments. All instances for training, validation, and testing are randomly chosen. The best and average error rates are listed and discussed.

7.4.1 Choosing the Training-Validation-Test Data Ratio

The experiment uses 3 different ratios of data, 800:100:100, 900:50:50 and 600:200:200 to determine the most suitable one. To be more accurate, all groups of data are chosen randomly from the German dataset. The number of hidden neurons varies from 6 to 39, which ensures that every group can get their best model. Additionally, each kind of model is trained 20 times. This section records the lowest error rate and average rate of each kind of model. Test results are listed in Table 6.

Table 6 Test results and comparison of different ratios of data

	800:100:100		900:50:50		600:200:200	
Number of hidden neurons	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate
6	0.21	0.2415	0.21	0.2925	0.245	0.273
7	0.21	0.251	0.2	0.251	0.225	0.273
8	0.19	0.239	0.21	0.2765	0.22	0.2615
9	0.18	0.2385	0.18	0.2315	0.245	0.27675
10	0.18	0.2295	0.2	0.243	0.24	0.27075
11	0.2	0.242	0.18	0.2635	0.225	0.26275
12	0.21	0.246	0.19	0.2675	0.23	0.26575
13	0.2	0.256	0.21	0.2705	0.2	0.26625
14	0.21	0.258	0.19	0.258	0.205	0.26525
15	0.17	0.2595	0.21	0.26	0.24	0.2715
16	0.22	0.246	0.23	0.26	0.23	0.266
17	0.19	0.248	0.21	0.245	0.235	0.2745
18	0.21	0.2595	0.21	0.293	0.21	0.26725
19	0.2	0.255	0.24	0.277	0.235	0.28
20	0.21	0.2585	0.21	0.262	0.22	0.2655
21	0.22	0.256	0.21	0.2835	0.24	0.2665
22	0.2	0.2545	0.22	0.2985	0.23	0.26475
23	0.19	0.259	0.21	0.2675	0.24	0.27075
24	0.19	0.26	0.2	0.26	0.245	0.27375
25	0.22	0.257	0.21	0.2845	0.24	0.27175

26	0.22	0.2725	0.19	0.276	0.235	0.28575
27	0.2	0.261	0.22	0.287	0.24	0.2735
28	0.21	0.2625	0.23	0.2755	0.235	0.2755
29	0.21	0.262	0.19	0.266	0.225	0.273
30	0.23	0.2765	0.23	0.278	0.21	0.2665
31	0.24	0.2715	0.23	0.2775	0.25	0.28575
32	0.22	0.265	0.2	0.2775	0.245	0.272
33	0.23	0.2755	0.22	0.273	0.245	0.27725
34	0.22	0.272	0.22	0.281	0.24	0.28575
35	0.22	0.2675	0.22	0.2885	0.235	0.28225
36	0.24	0.2835	0.19	0.285	0.245	0.2895
37	0.22	0.278	0.23	0.3175	0.225	0.29125
38	0.17	0.28	0.22	0.301	0.21	0.27825
39	0.24	0.294	0.2	0.284	0.235	0.284
Best	0.17	0.2295	0.18	0.2315	0.2	0.2615
Average	0.208235	0.259882	0.209412	0.273897	0.231618	0.27375

The results show that a ratio of 800:100:100 performs better in nearly all aspects in regards to accuracy rate. The lowest error rate, 0.17, is achieved with 15 hidden units. This is also the best model of all. The average error rate can indicate an overall performance of some model groups. The model with 10 hidden units seems more stable, with an average error rate of 0.2295, which is lower than the others. For all models, the average lowest error rate is around 0.208, indicating that it is more likely to get a very low error rate with this training-validation-test ratio.

The ratio of 600:200:200 gives more data to testing, which leads to insufficient data for training. Thus it gets high error rates in regards to both the lowest value and the average value. Although the ratio 900:50:50 has more instances for training and the lowest error rate is very close to the best one, the shortness of testing data leads to a high average value, which means that there is a low possibility to get a good model.

7.4.2 Training with ARCM

This section compares the models trained with data from ARCM. Nothing has changed except the percentage of approved/rejected instances in each dataset. The overall percentage is 70% for approved instances and 30% for rejected instances. So this ratio

stays the same in the training, validation, and test data groups. The ratio of training-validation-test is 800:200:200, which perform best in the previous tests.

To compare ARCM with other benchmark instance selection algorithms, SMOTE is used as a baseline model. The algorithm of SMOTE is introduced in section 2.3.1. In this experiment, SMOTE is applied in Weka. The number of nearest neighbours to use is set to 5 (default in Weka). The new dataset generated by SMOTE is then divided into training, validation and test groups with a ratio of 8:2:2. It should be noticed that as SMOTE generate new instances, the number of instances in each group will not be the same as those in ARCM test. The potential side-effect brought by this change is ignored in this research.

The results are listed in Table 7.

Table 7 Test results and comparison of two instance choosing methods

Number of hidden neurons	ARCM				SMOTE		Pure random choosing method	
	Lowest test error rate	Average error rate	lowest validation error rate	Average validation error rate	Lowest test error rate	Average error rate	Lowest test error rate	Average error rate
6	0.14	0.2145	0.1265	0.1502	0.20	0.223	0.21	0.2415
7	0.15	0.211	0.1228	0.1441	0.19	0.2155	0.21	0.251
8	0.17	0.2215	0.1231	0.1438	0.17	0.209	0.19	0.239
9	0.13	0.2085	0.1284	0.1472	0.18	0.2235	0.18	0.2385
10	0.13	0.221	0.1270	0.1471	0.19	0.252	0.18	0.2295
11	0.16	0.223	0.1234	0.1439	0.18	0.215	0.2	0.242
12	0.13	0.225	0.1297	0.1491	0.18	0.2255	0.21	0.246
13	0.16	0.2225	0.1163	0.1431	0.16	0.242	0.2	0.256
14	0.14	0.2105	0.1113	0.1407	0.19	0.234	0.21	0.258
15	0.16	0.219	0.1103	0.1422	0.18	0.2485	0.17	0.2595
16	0.14	0.2085	0.1245	0.1399	0.21	0.2275	0.22	0.246
17	0.19	0.2415	0.1196	0.1433	0.19	0.226	0.19	0.248
18	0.17	0.221	0.1022	0.1342	0.19	0.2195	0.21	0.2595
19	0.14	0.2135	0.1124	0.1407	0.20	0.2365	0.2	0.255
20	0.16	0.2215	0.1006	0.1402	0.16	0.237	0.21	0.2585
21	0.16	0.215	0.1207	0.1368	0.17	0.2335	0.22	0.256
22	0.15	0.2195	0.1217	0.1351	0.17	0.248	0.2	0.2545

23	0.18	0.2265	0.1143	0.1353	0.16	0.2185	0.19	0.259
24	0.14	0.215	0.1249	0.1377	0.19	0.25	0.19	0.26
25	0.15	0.223	0.1007	0.1394	0.17	0.2545	0.22	0.257
26	0.14	0.224	0.0884	0.1310	0.17	0.228	0.22	0.2725
27	0.17	0.2175	0.1227	0.1380	0.16	0.2245	0.2	0.261
28	0.15	0.2235	0.1123	0.1345	0.19	0.2285	0.21	0.2625
29	0.15	0.2265	0.1100	0.1339	0.15	0.231	0.21	0.262
30	0.15	0.2275	0.0916	0.1304	0.16	0.238	0.23	0.2765
31	0.18	0.228	0.0892	0.1283	0.16	0.253	0.24	0.2715
32	0.18	0.234	0.1013	0.1295	0.19	0.2585	0.22	0.265
33	0.17	0.2345	0.1070	0.1287	0.21	0.2515	0.23	0.2755
34	0.17	0.228	0.1023	0.1285	0.22	0.2275	0.22	0.272
35	0.14	0.229	0.0932	0.1243	0.19	0.236	0.22	0.2675
36	0.14	0.2275	0.1055	0.1311	0.17	0.2165	0.24	0.2835
37	0.16	0.2265	0.0963	0.1258	0.17	0.2205	0.22	0.278
38	0.19	0.2385	0.0801	0.1324	0.18	0.218	0.17	0.28
39	0.14	0.2375	0.0854	0.1215	0.20	0.221	0.24	0.294
Best	**0.13**	**0.2085**	**0.0801**	**0.1215**	**0.15**	**0.215**	**0.17**	**0.2295**
Average	**0.1553**	**0.2231**	**0.1102**	**0.1368**	**0.181**	**0.2321**	**0.2082**	**0.2599**

Comparing with the best and the average error rates in Table 6, the results from this round of testing are clearly better. The lowest error rate is 0.13, which is 0.04 lower than the previous experiments. The average value of the lowest error rate is around 0.155, which is a 25% improvement (the previous best rate was 0.208). This indicates that, with this kind of data, it is more likely to get a high accuracy model that has high predictability.

Comparing with SMOTE and pure random choosing, ARCM has some advantages in terms of the prediction accuracy. SMOTE is not as good as ARCM because of the instances generated during this approach may bring noise to the dataset. As a matter of fact, ARCM is designed to be more suitable for datasets with slightly imbalance, while SMOTE is designed for highly imbalanced datasets.

To be more objective, this thesis uses another credit rating dataset to validate the performance of our method. The Australian credit dataset contains 690 instances and 15 attributes. 307 of all instances belong to class 1 and the other 383 instances belong to

class 2. Thus, this credit rating dataset is not highly imbalanced. A similar training method is applied to this dataset. However, as this dataset only contains 15 attributes, which is less than the German credit dataset, the number of hidden nodes is set from 3 to 32. Test results are listed in Table 8.

Table 8 Test results and comparison of two instance choosing methods with the Australian credit dataset

	ARCM		Pure random choosing method	
Number of hidden neurons	Average Test Error	Average Validation Error	Average Test Error	Average Validation Error
3	0.1406	0.1315	0.1290	0.1239
4	0.1292	0.1329	0.1486	0.1343
5	0.1389	0.1286	0.1601	0.1258
6	0.1341	0.1314	0.1420	0.1349
7	0.1401	0.1253	0.1406	0.1252
8	0.1570	0.1196	0.1355	0.1221
9	0.1534	0.1373	0.1333	0.1283
10	0.1377	0.1133	0.1355	0.1233
11	0.1304	0.1286	0.1428	0.1216
12	0.1522	0.1209	0.1543	0.1162
13	0.1304	0.1165	0.1399	0.1248
14	0.1401	0.1271	0.1630	0.1188
15	0.1268	0.1267	0.1370	0.1191
16	0.1196	0.1142	0.1449	0.1194
17	0.1292	0.1157	0.1377	0.1193
18	0.1510	0.1265	0.1312	0.1184
19	0.1353	0.1227	0.1536	0.1171
20	0.1280	0.1204	0.1522	0.1148
21	0.1147	0.1170	0.1543	0.1134
22	0.1220	0.1201	0.1304	0.1181
23	0.1437	0.1284	0.1464	0.1118
24	0.1498	0.1265	0.1384	0.1159
25	0.1449	0.1273	0.1536	0.1163
26	0.1425	0.1284	0.1522	0.1146
27	0.1413	0.1182	0.1464	0.1151
28	0.1437	0.1169	0.1377	0.1105
29	0.1341	0.1182	0.1304	0.1081

30	0.1220	0.1099	0.1464	0.1082
31	0.1135	0.1055	0.1601	0.1072
32	0.1389	0.1119	0.1399	0.1103
Best	**0.1135**	**0.1055**	**0.1290**	**0.1072**
Average STD	**0.0384**	**0.0139**	**0.0389**	**0.0142**

Test results indicate that the novel method can also improve the performance of credit rating with the Australian credit dataset. The error rate declines from 12.9% to 11.3% with test data and 10.7% to 10.5% in validation. This test proves that the ARCM can still optimize the training of neural networks even when the dataset is not highly imbalanced.

Test results also show that better organized data can enhance model performance, especially when the dataset is imbalanced between its classes. As the German dataset is a real world application, ARCM can be easily generalised to handling other credit datasets. However, when the dataset is not highly imbalanced, the new method can also improve the performance to some extent.

7.4.3 Number of Hidden Neurons

In Table 6, this thesis also lists the result of validation for each kind of model, including the lowest and average rates among the 20 different models. Regarding the test data error rates, the lowest ones seem to have been achieved when the number of hidden neurons is 9, 10, or 12, respectively. However, with these numbers, many other models also get competitive results. When the number of hidden units is 6, 14, 16, 19, 24, 26, 35, 36, and 39, respectively, the lowest rate is 0.14, which is only a little higher than 0.13 so they can still be regarded as good models. As to the average rate, the model with 9 hidden neurons gets the lowest one, which is 0.2085. The average of all models is only 0.223, which means that there is little difference between models when only accuracy is considered. This means that there is no ubiquitous principle for the relationship between the number of hidden neurons and the accuracy of a model.

However, computation time is also very important to neural network research. More hidden neurons can lead to more computation time. Thus, if some models produce the same accuracy, the one with less hidden neurons is preferred. In our experiments, the

network with 9 hidden neurons wins in most cases. As such, this model is chosen as the most suitable for the German credit dataset in the experiments.

Table 9 Test results of the best model with highest accuracy and efficiency

	800:100:100		900:50:50		600:200:200		ARCM	
	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate	Lowest error rate	Average error rate
9 hidden neurons	0.18	0.2385	0.18	0.2315	0.245	0.2768	0.13	0.2085
Best of all	0.17	0.2295	0.18	0.2315	0.2	0.2615	0.13	0.2085
Average	0.2082	0.2599	0.2094	0.2739	0.2316	0.2738	0.1553	0.2230

It is more interesting when checking the validation error rates. As validation data is also a kind of test data (but with a different purpose), the error rate can also reflect predictability. It is always higher than the accuracy of test data because a training process will not stop until a validation gets high accuracy. In this experiment, the accuracy of validation can reach almost 0.92 with 38 hidden units in the model. This is 0.05 higher than the best result achieved with the test data. Although this value is not as objective as the accuracy from pure test data, it indicates that the MLP model has the ability of rating credit applications more precisely. Also, there is an interesting tendency found in the validation dataset. As the number of hidden units gets larger, the error rate seems to get lower. This is shown in Figure 14.

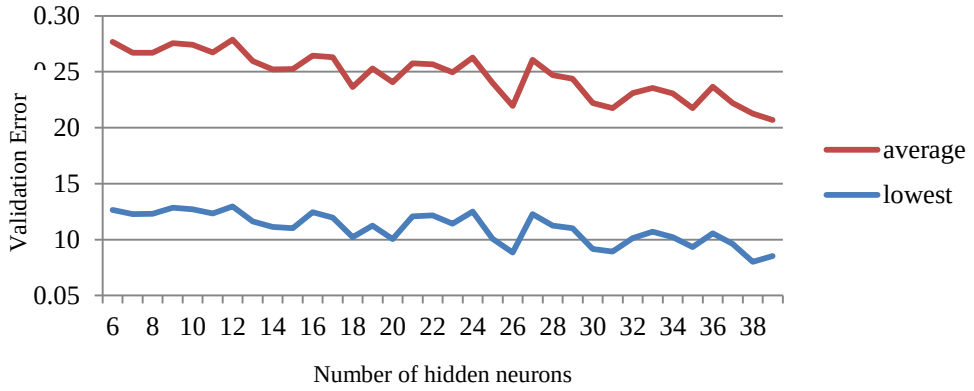

Figure 14 Tendency of the model's error rates when the number of hidden units increases

7.4.4 Summary of ARCM Experiments

In the first round of experiments, the experiment compares the accuracies of models trained with different ratios of training-validation-test data. The 800:100:100

combinations get the highest accuracy, so this ratio is most suitable for building an acceptable model. After that, this thesis use ARCM to optimize the dataset. The ratio of approved/rejected instances in the German dataset, 7:3, is precisely implemented in the datasets for training, validation, and testing. Results show that the new method can remarkably enhance the accuracy, from 83% to 87% for the best model. For those datasets that are not extremely imbalanced, ARCM has some advantages when compared with SMOTE. Finally, models with 9 hidden units perform best out of all models in terms of high accuracy and low computational time. Also, this thesis finds an interesting relationship between the number of hidden neurons and validation accuracy: the more hidden units a model has, the higher the accuracy it may get when validated.

Compared with the results in other relevant articles with the same benchmark dataset in Table 10, the new model achieves a high accuracy of 87%, which is almost higher by 5% than the best result reported in the relevant literature so far. The best model contains 9 hidden neurons, using ARCM. The ratio of training-validation-test data is 800:100:100. If taking validation results into consideration, the highest accuracy reaches 92%, which is almost 10% higher than the best result from existing models.

Table 10 Prediction accuracies found in other researches

Article name	Scoring Models	Accuracy (%)
(Marcano-Cedeño et al., 2011)	MLP	84.67±1.5
(Xiao et al., 2012)	Ensemble	82.03
(Brown and Mues, 2012)	LS-SVM	81.9
(Khashei et al., 2013)	MLP	81.3
(Khashman, 2011)	MLP	81.03
(Hens and Tiwari, 2012)	SVM	80.42
(Wang et al., 2012)	DT	78.52
(Setiono et al., 2011)	Re-Rx	78.47
(Yu et al., 2011)	SVM	78.46
(Vukovic et al., 2012)	Case-based reasoning model	77.4
(Ping and Yongheng, 2011)	SVM	76.6
(Gonen et al., 2012)	SVM	75.4
(Marqués et al., 2013)	SVM	71.8

7.5 Experiments and Results of Novel Filter CCBFS

In this section the experiment settings and test results of CCBFS will be demonstrated, along with the explanations.

As filters take no consideration of machine learning algorithms, good filters should perform well on different kinds of learning algorithms. Here three categories of learning algorithms are used, including ANN, decision tree and Naïve Bayes. Tests on these different learning algorithms will demonstrate the performance of filters from more aspects.

To prove that CCBFS has better performance than those compared filters, two key measures will be shown: the AUC of machine learning, and the number of selected features.

As this research need to use CCBFS as a part of the novel hybrid feature selection methods, its performance on ANN is more important and will be tested on three different ANN models: MLP, RBF and HONN. All these machine learning models used in this chapter have been introduced in Chapter 2.

The goal of this section is to prove that CCBFS is a better choice to be used in the hybrid feature selection algorithm, as it can select out fewer features with better performance.

7.5.1 Experiment Setup

In this experiment, five representative feature selection algorithms will be compared with CCBFS. They include Liu's Consistency based Feature Selection (ConsistencyFS), Hall's Correlation-based Feature Selection (CorrelationFS), Mutual Information based feature selection (GainRatio), Zhao & Liu's interaction based feature selection (INTERACT) and ReliefF. The original datasets are also used to demonstrate the effect of feature selection. All these feature selection algorithms, except INTERACT, can be found in WEKA. The java code of INTERACT algorithm can be found http://machinelearningresearch.googlecode.com/svn/trunk/MachineLearning/src/weka/attributeSelection/INTERACT.java, provided by Zhao et.al. Thus, all experiments are conducted under WEKA environment.

To verify the effective of CCBFS, this experiment is based on many kinds of data. All these datasets are introduced in the section 7.2. This experiment uses default settings in WEKA for MLP, C4.5 and Naïve Bayes. As to MLP, it uses two hidden layers and the training time is set to 5000. The other settings of MLP are default values in WEKA. All the AUCs are the mean value of 10 times training, and each training uses 10 fold cross validation with random split seeds.

In this test, feature ranking filters such as CCBFS, ReliefF and GainRatio are not able to product feature subsets which can be compared. So a forward searching will be applied to these three filters and the top *k* features of the ranking list that can produce the highest AUC will be selected. A more objective comparison between feature ranking methods is the AUC line drawn by the forward search.

7.5.2 Feature Subset Selection Results

In this test, MLP will be used as an ANN model as it is prevalently used in machine learning. The setting command of MLP in Weka is as follows:

```
L 0.3 -M 0.2 -N 5000 -V 10 -S 0 -E 20 -H a -B
```

To be more specific, the MLP will contain one hidden layer and the number of neurons is calculated by (attributes + classes)/2. Learning rate is 0.3 and momentum is 0.2. The maximum training time is set as 5000. Validation size is 10% of the whole dataset. Validation threshold is 20 so the training will be terminated if the validation set error gets worse for 20 times. All other settings are default values. It should be mention that each evaluating process will use different training, validation and test data group generated by ARCM. So although the setting of MLP stays the same, the test results may vary in each time. And that is the reason that 10 times training is used to get an average value (including standard deviation).

The number of selected features for each dataset is shown in Table 11, and AUCs of MLP are in Table 12.

Table 11 Number of selected features by filters

Dataset	Number of Features						
	CCBFS	CorrelationFS	ConsistencyFS	GainRatio	INTERACT	ReliefF	Ori(no FS)
Aus	6	8	12	11	12	15	15
Ve	19	12	19	19	19	19	19
Statlog	12	4	14	14	15	12	21
Io	13	18	11	19	9	9	35
Wf	19	16	13	21	13	18	41
Bio	34	16	20	35	12	33	42
Lc	5	9	7	5	7	5	57
So	15	20	13	28	13	38	61
Average	15.375	12.875	13.625	19	12.5	18.625	36.375

From Table 11 it can be seen that all feature selection algorithms have obvious effects of removing features, especially to those with more features. When looking at the average number of selected features, feature ranking methods (CCBFS, ReliefF and GainRatio) select out more features than feature subset selecting methods (ConsistencyFS, Correlation FS and INTERACT). The reason is that feature subset selection methods can exclude redundant features, while feature ranking methods cannot. Thus, there may be more redundant features in the selected subset for feature ranking methods.

Table 12 shows the AUC and std of MLP. The AUCs of CCBFS are marked bold if it is the highest (or one of the highest) for each dataset. It can be seen that CCBFS can reach highest AUC in 6 of the 8 data groups. In the Bio dataset, CCBFS loses GainRatio and ReliefF by only 0.01. The average AUC of all 8 datasets is 0.888 achieved by CCBFS, higher than all other filters.

It should be noticed that feature subset selection methods perform worse than feature ranking methods on the aspect of AUC. That is because the forward searching is actually a kind of wrapper methods. It will stop at the maximum AUC and return the selected features as the outputs. Although this advantage makes CCBFS perform better than ConsistencyFS, CorrelationFS and INTERACT, it can still prove that CCBFS is competitive with these three filters.

Table 12 Experimetn for CCBFS: MLP performances with different filters

	Filters														
data	CCBFS	std	Correlat ionFS	std	Consiste ncyFS	std	GainR atio	std	INTE RACT	std	Relief F	std	Ori(no FS)	std
Aus	**0.92**	0.03	0.92	0.03	0.915	0.03	0.918	0.032	0.916	0.03	0.914	0.032	0.914	0.033
Ve	**0.946**	0.058	0.891	0.067	0.946	0.058	0.946	0.058	0.946	0.058	0.946	0.058	0.946	0.059
Statlog	**0.742**	0.041	0.742	0.051	0.729	0.047	0.742	0.045	0.734	0.048	0.736	0.049	0.726	0.044
Io	**0.943**	0.066	0.943	0.06	0.926	0.066	0.943	0.065	0.917	0.068	0.934	0.048	0.903	0.072
Wf	**0.973**	0.006	0.973	0.007	0.961	0.009	0.973	0.007	0.969	0.01	0.973	0.007	0.97	0.007
bio	0.909	0.037	0.885	0.036	0.895	0.033	0.91	0.029	0.899	0.035	0.91	0.03	0.908	0.031
LC	**0.799**	0.046	0.719	0.055	0.745	0.05	0.779	0.043	0.744	0.057	0.799	0.044	0.662	0.061
So	0.872	0.052	0.864	0.057	0.833	0.062	0.876	0.05	0.818	0.061	0.871	0.054	0.849	0.057
Average	**0.888**		0.867		0.869		0.886		0.868		0.885		0.860	

When looking into the details, there are some datasets that filters has no effect. For example, in Vehicle dataset, the AUC of original dataset is the same as CCBFS and some other filters. 5 of 6 filters cannot reduce feature subset, which indicates that these filters can find no irrelevant or redundant features in Vehicle.

The Waveform dataset contains 41 features, and 19 of them have already been marked as noise attributes. It is clear to see how the irrelevant attributes can hinder the performance of machine learning. All filters eliminate those noise attributes and 5 of them improved the AUC. These improvements take the advantage of fewer but more relevant features in training dataset.

In the end, CCBFS also shows good performance on all kinds of data: small (Aus, Ve), medium (Statlog, Io, Wf, Bio) and large (So, Lc); and with different categories of real world problem: binary classification (Aus, Statlog, Io, Bio, So) and multi class recognition (Ve, Wf, Lc).

This result indicates that CCBFS is competitive with other benchmark feature selection algorithms on the aspect of improving performance of ANN. The selected feature subsets by CCBFS have higher AUC than both original dataset and some of the filters. Thus, CCBFS has been proved to be capable to be used as a feature pre selection in the hybrid feature selection process.

7.5.3 Feature Ranking Performance

To objectively compare the performance of CCBFS with other feature ranking filters, this experiment records the AUCs of the forward subset searching process.

After getting a feature ranking list, the highest ranked feature is added into the feature subset each time and records the performance of MLP trained by that subset. The lines in each figure represent the variance of AUC in this searching process.

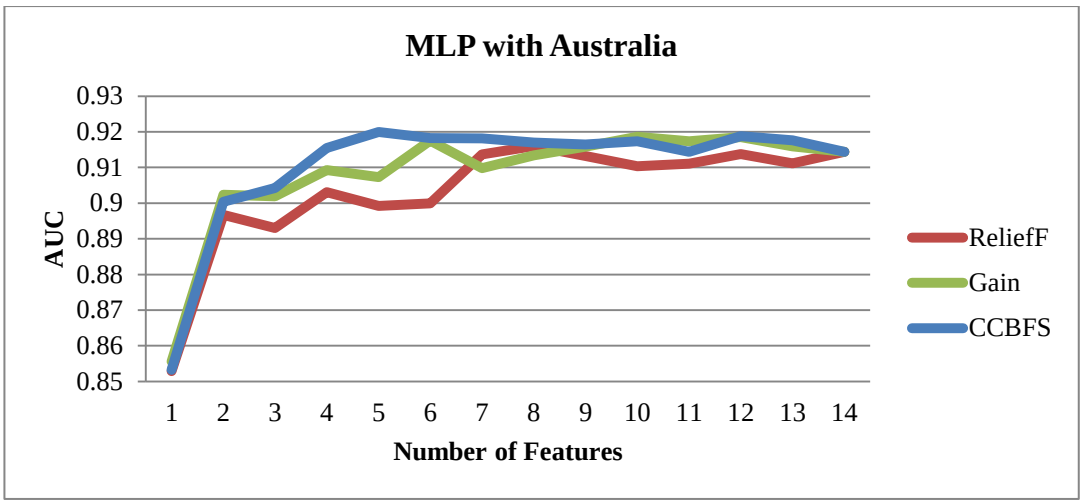

Figure 15 Feature ranking performance under MLP with Australia dataset

In the test of Australia credit dataset, CCBFS has obvious advantage comparing with the other two filters, as it reaches the maximum earlier (which means CCBFS can achieve higher AUC with less features). There is no obvious decrease made by irrelevant features, as the AUCs do not drop a lot in the end.

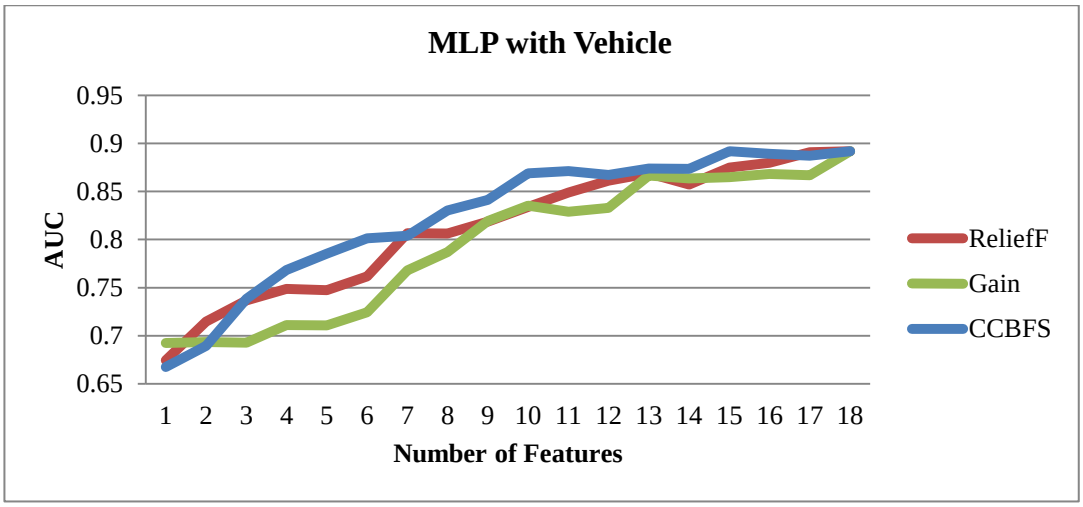

Figure 16 Feature ranking performance under MLP with Vehicle dataset

In the test of Vehicle dataset, CCBFS outperforms the other two filters as it always has higher AUC with the same number of features. It should be noticed that all three lines keep rising, which indicates that there may be very few irrelevant features. However, redundant features may exist, as in some areas there are no obvious increasing with more features.

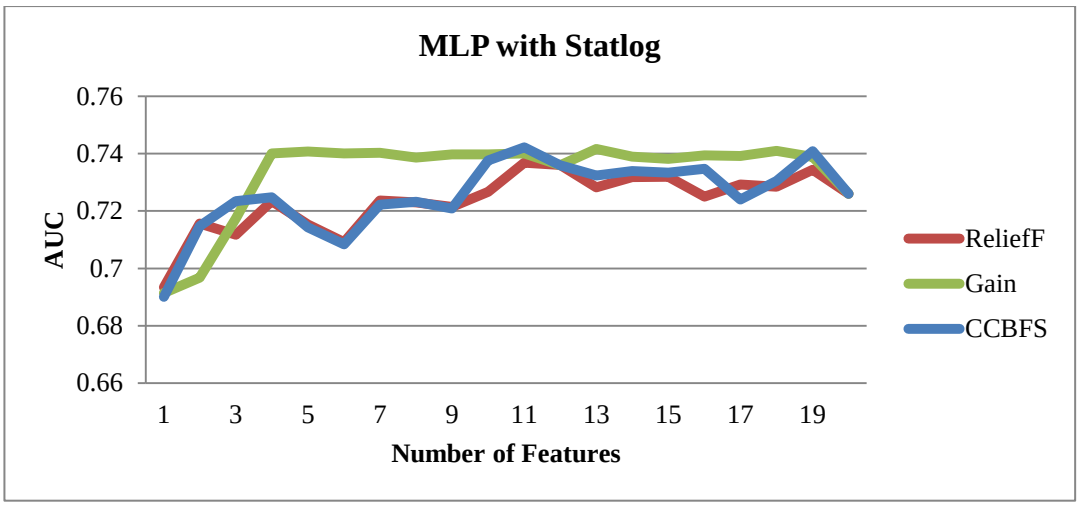

Figure 17 Feature ranking performance under MLP with Statlog dataset

In the test of Statlog dataset, GainRatio performs better as it reaches high leval with fewer features. Also it should be mentioned that there is no obvious increasing of AUC when the number of features increase. This means there are lots of redundant features in this dataset.

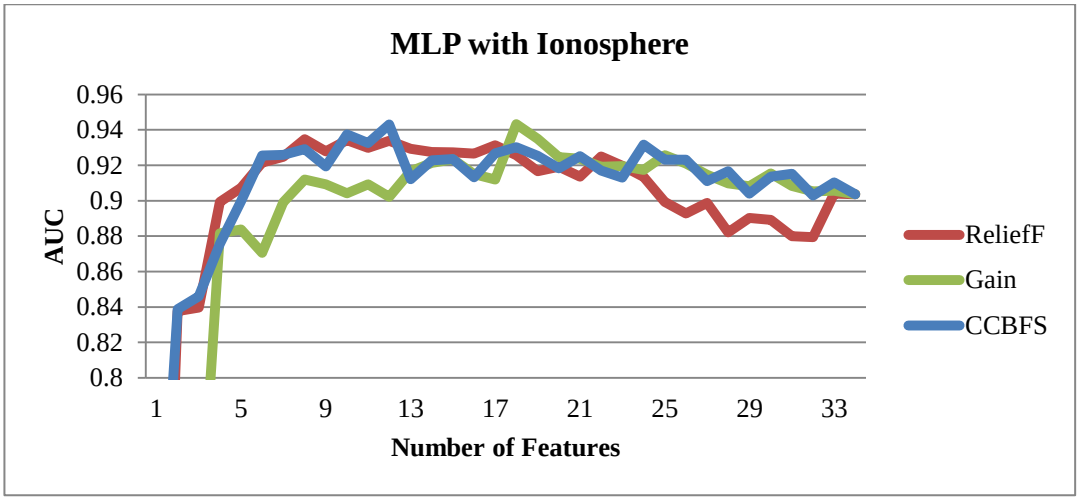

Figure 18 Feature ranking performance under MLP with Ionosphere dataset

In the test of Ionosphere dataset, all three filters have similar performances, with GainRatio slightly worse than the other two. All three lines have tendencies of decreasing after about 13 features.

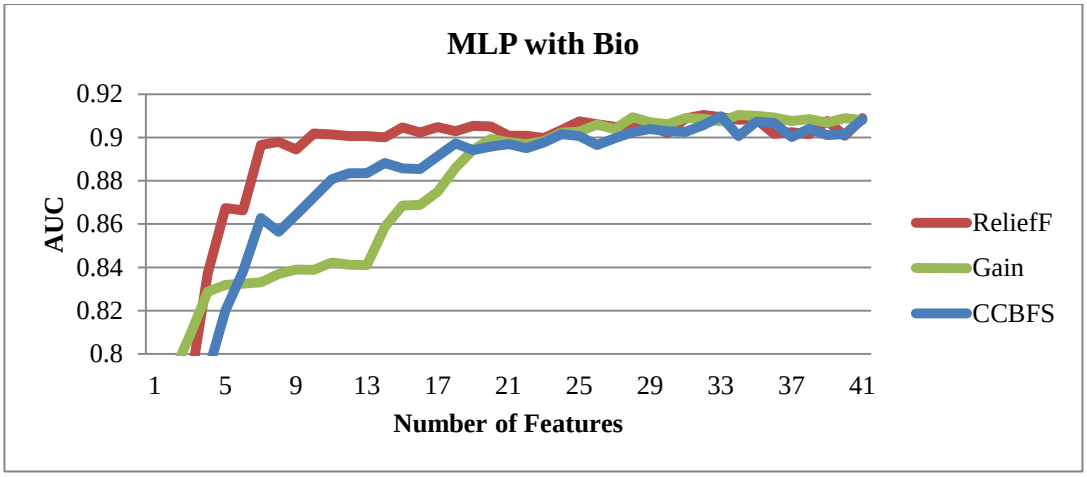

Figure 19 Feature ranking performance under MLP with Bio dataset

In the test of Bio dataset, CCBFS performs better than GainRatio, but worse than ReliefF. The AUCs of all three methods keep increasing to the end.

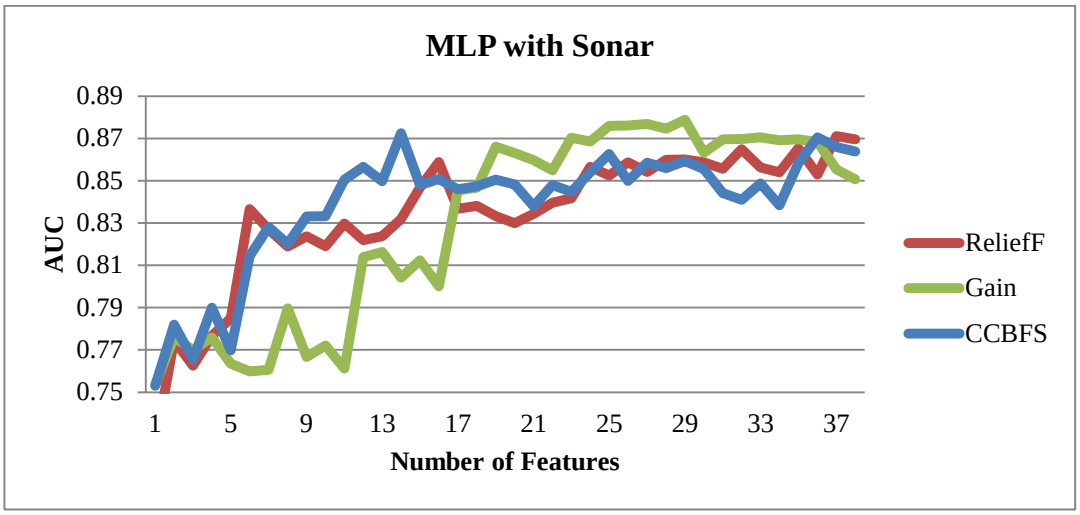

Figure 20 Feature ranking performance under MLP with Sonar dataset

In the test of Sonar dataset, CCBFS has little advantage to ReliefF and GainRatio at the area between 8 and 15. It reaches maximum earlier than the other two, but performs worse when adding more features into the selected group.

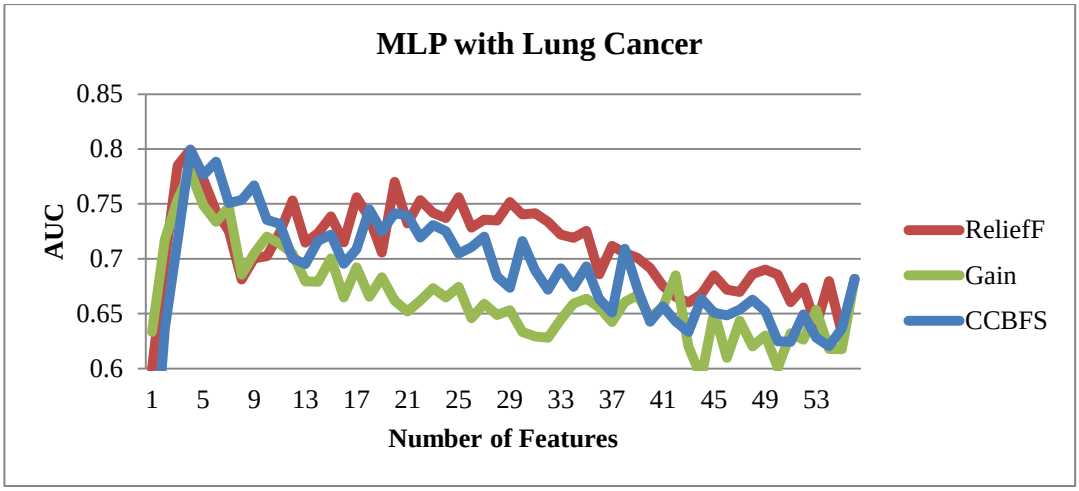

Figure 21 Feature ranking performance under MLP with Lung Cancer dataset

In the test of Lung Cancer dataset, all three filters get their top very early at around 5. Then they decrease somehow, which indicates there may be may irrelevant features in this dataset. CCBFS and ReliefF perform a little bit better than GainRatio this time.

From this experiment it is clear to see that CCBFS has advantages in some dataset, while the other two feature ranking methods may be more suitable for others. In general, CCBFS is a competitive feature ranking method, and has proved to be capable to be used in assisting the local optimization of hybrid feature selection process.

7.5.4 CCBFS Performance Under Other Machine Learning Algorithms

As a filter process, CCBFS is not just designed for one particular learning algorithm. The performance of CCBFS should be evaluated under more categories of learning algorithms. In this research, decision tree (J48 tree) and Naïve Bayes network will also be used to test the ability of CCBFS.

7.5.4.1 Decision Tree Feature Subset Selection Results

This test uses an unpruned C4 tree as the machine learning algorithm. The default settings of WEKA are used in each test (confidence factor 0.25). The test result is shown in Table 13 and Table 14.

Similarly, from Table 13 it can be seen that all feature selection algorithms remarkably reduce the number of features. ReliefF has the least number of average selected features and CCBFS has the second least.

Table 13 Number of selected features by filters

Data	Num of Features						
	CCBFS	CorrelationFS	ConsistencyFS	GainRatio	INTERACT	ReliefF	Ori(no FS)
Aus	7	8	12	7	12	8	15
Ve	10	12	19	14	19	17	19
Statlog	4	4	14	4	15	3	21
Io	23	18	11	32	9	14	35
Wf	11	16	13	9	13	7	41
Bio	12	16	20	20	12	7	42
Lc	7	9	7	6	7	6	57
So	22	20	13	20	13	8	61
Average	12	12.875	13.625	14	12.5	8.75	36.375

As to the classification AUC, CCBFS again has the highest average AUC (0.836), with RliefF slightly lower (0.835). GainRatio is also higher than those 3 feature subset selection methods (CorrelationFS 0.818, ConsistencyFS 0.806 and INTERACT 0.816). The reason that “filter ranking + forward selection” performs better has been explained in the last section.

The test result of Vehicle dataset in this experiment is similar with MLP experiment. Again the reason that filters have no obvious improvement is there are very few irrelevant features in this dataset.

There are three dataset in this test that CCBFS loses ReliefF (Wf, Lc and So). This result indicates that ReliefF performs better with Decision Tree than with MLP, and CCBFS fits for MLP better than Decision Tree.

Table 14 AUC of J48 with filters

data	Filters														Ori(no FS)	std
	CCBFS	std	CorrelationFS	std	ConsistencyFS	std	GainRatio	std	INTERACT	std	ReliefF	std				
Aus	**0.898**	0.041	0.883	0.04	0.87	0.05	0.888	0.045	0.866	0.049	0.883	0.041			0.862	0.052
Ve	**0.857**	0.051	0.856	0.052	0.854	0.061	0.857	0.055	0.854	0.06	0.857	0.049			0.854	0.058
Statlog	**0.715**	0.057	0.713	0.048	0.679	0.056	0.715	0.055	0.669	0.054	0.715	0.049			0.647	0.061
Io	**0.915**	0.058	0.907	0.068	0.909	0.055	0.898	0.064	0.903	0.059	0.913	0.058			0.891	0.06
Wf	0.868	0.02	0.845	0.02	0.857	0.02	0.87	0.017	0.859	0.02	0.885	0.018			0.829	0.025
Bio	**0.868**	0.049	0.842	0.05	0.829	0.043	0.834	0.043	0.842	0.049	0.853	0.05			0.822	0.041
Lc	0.750	0.032	0.695	0.039	0.743	0.041	0.751	0.030	0.723	0.031	0.757	0.035			0.586	0.064
So	0.819	0.026	0.801	0.028	0.705	0.032	0.81	0.028	0.814	0.027	0.82	0.025			0.754	0.038
Average	**0.836**		0.818		0.806		0.828		0.816		0.835				0.781	

7.5.4.2 Decision Tree Feature Ranking Performance

In this part an objective shown of performance of CCBFS will be displayed by comparisons between three feature ranking methods.

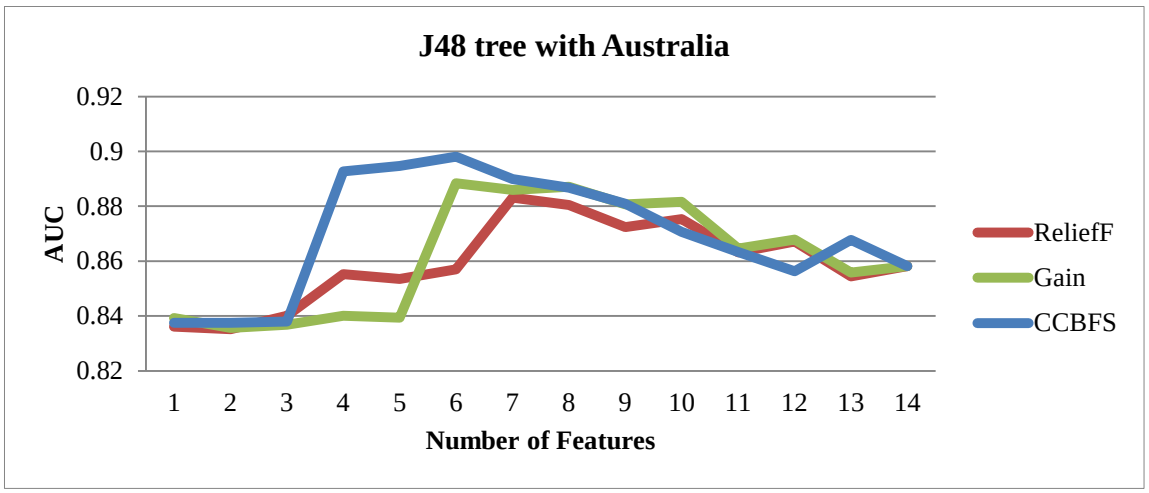

Figure 22 Feature ranking performance under J48 with Australia dataset

CCBFS performs better than the other two filter ranking methods with Australia dataset, as it reaches higher AUC and with fewer attributes. All three lines have obvious declines in the end, which indicates there are some irrelevant features drawing back the performance in the original dataset.

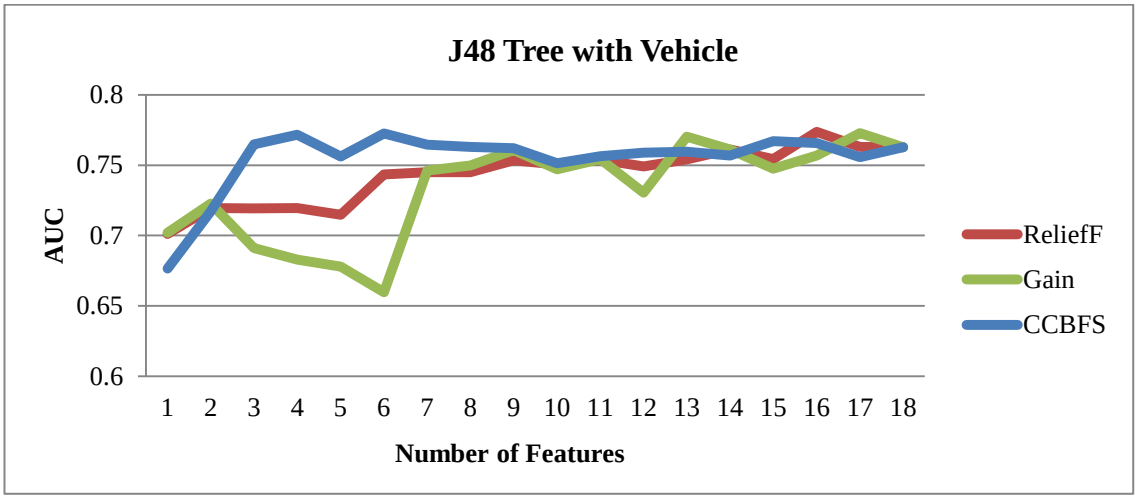

Figure 23 Feature ranking performance under J48 with Vehicle dataset

Again CCBFS performs better in the test with Vehicle dataset. This dataset may contain many redundant features, but these redundant features will not affect the performance.

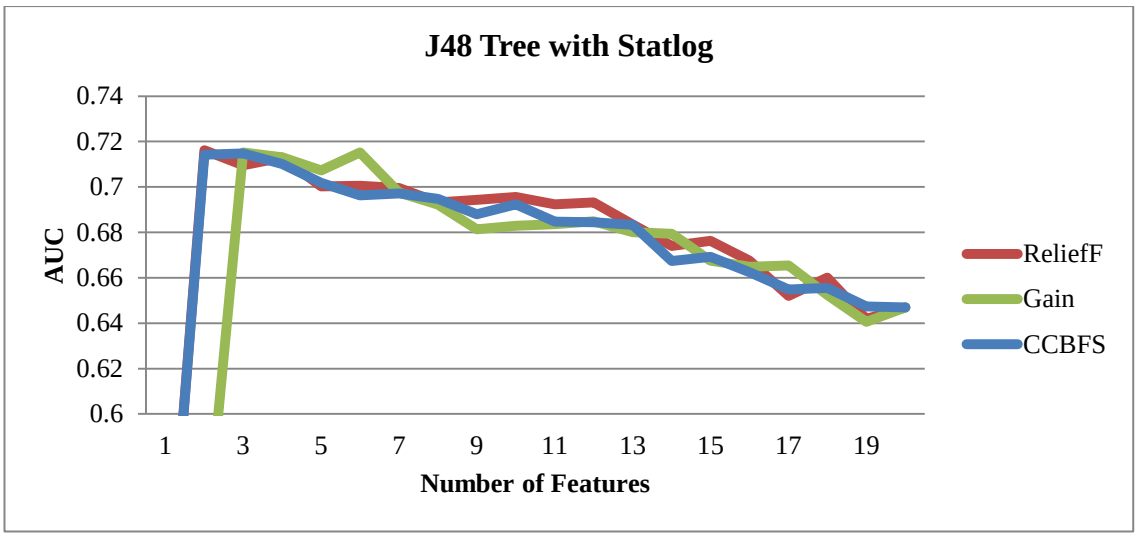

Figure 24 Feature ranking performance under J48 with Statlog dataset

All three filters reach their tops very early in the test with Statalog dataset. No one has obvious advantages in this test.

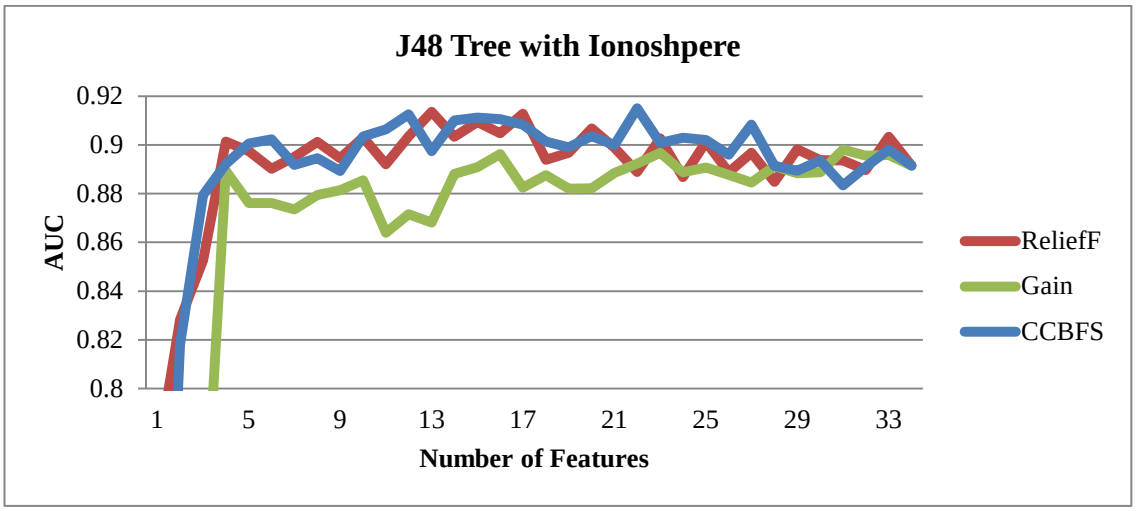

Figure 25 Feature ranking performance under J48 with Ionosphere dataset

In the test with Ionosphere dataset, CCBFS and ReliefF have similar performances and better than GainRatio, from the aspect of both AUC value and number of selected features.

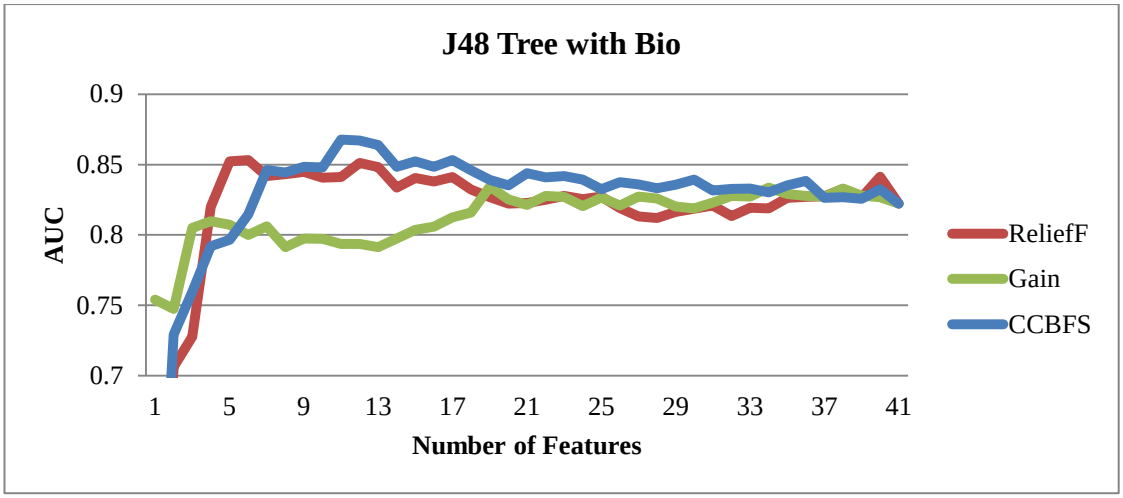

Figure 26 Feature ranking performance under J48 with Bio dataset

Similar with the last test, GainRation again performs a litter worse than the other two. While ReliefF performs better with fewer features, CCBFS has the highest AUC.

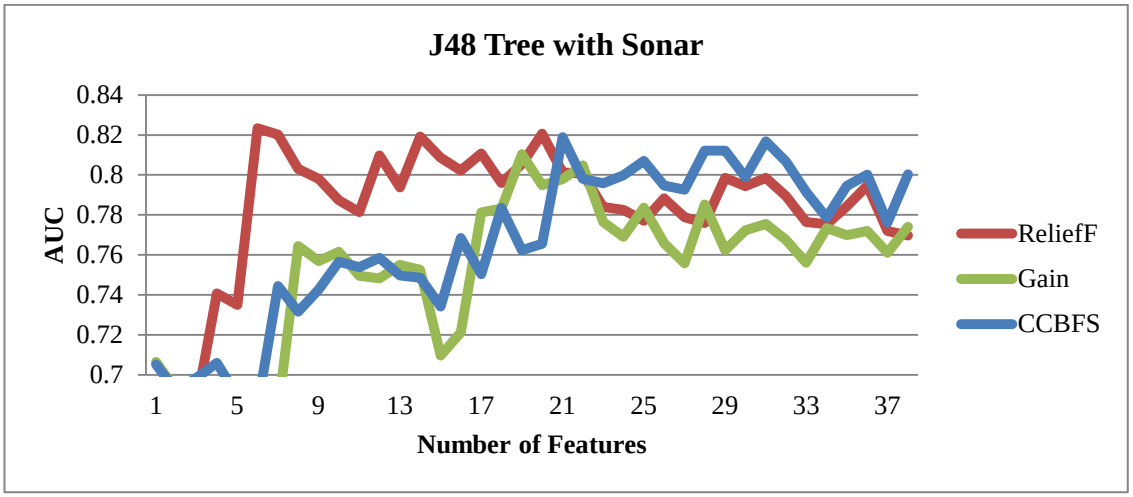

Figure 27 Feature ranking performance under J48 with Sonar dataset

This is one of the tests that ReliefF performs obviously better than others. It reaches the maximum with only few features, while the other two methods need more features.

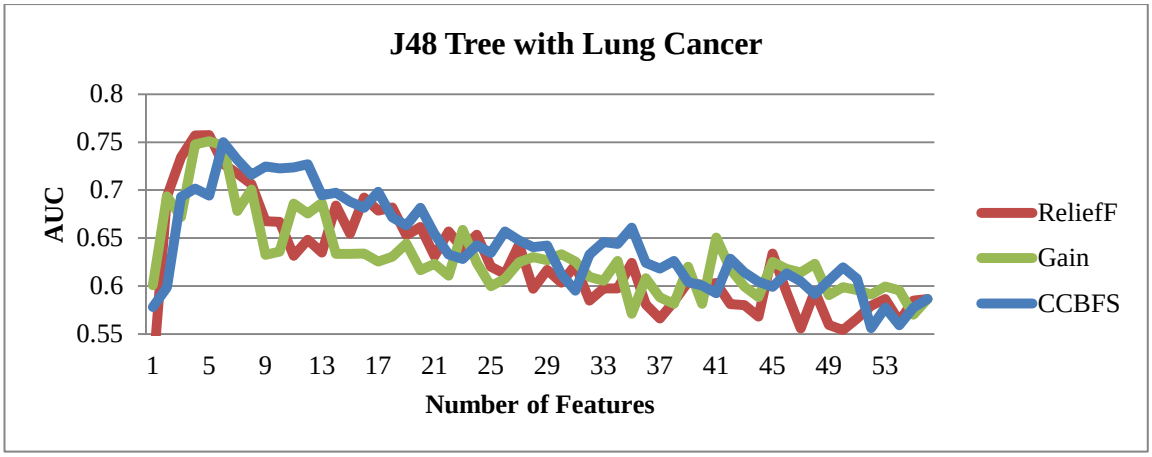

Figure 28 Feature ranking performance under J48 with Lung Cancer dataset

In the test with Lung Cancer dataset, all three filters have competitive performances. ReliefF is slightly better than others when checking each value. The declining tendencies indicate the existing of irrelevant features in this dataset.

7.5.4.3 Naïve Bayes Feature Subset Selection Results

This test uses Naïve Bayes in Weka as the learning algorithm. The default setting in Weka is used in each test. Numbers of selected features are in Table 15, and the AUCs and Stds are in Table 16.

Table 15 Number of selected features by filters

	Num of Features						
	CCBFS	CorrelationFS	ConsistencyFS	GainRatio	INTERACT	ReliefF	Ori(no FS)
Aus	7	8	12	8	12	8	15
Ve	16	12	19	18	19	18	19
Statlog	11	4	14	19	15	16	21
Io	12	18	11	23	9	13	35
Wf	14	16	13	14	13	13	41
Bio	25	16	20	30	12	38	42
Lc	19	9	7	13	7	23	57
So	22	20	13	26	13	36	61
Average	15.75	12.875	13.625	18.875	12.5	20.625	36.375

Table 16 AUC of Naïve Bayes with filters

	Filters													
Data	CCBFS	std	Correlation FS	std	Consistency FS	std	GainRatio	std	INTERACT	std	ReliefF	std	Ori(no FS)	std
Aus	**0.917**	0.033	0.902	0.031	0.894	0.036	0.900	0.036	0.893	0.036	0.917	0.031	0.875	0.036
Ve	**0.778**	0.048	0.755	0.051	0.766	0.052	0.767	0.053	0.766	0.051	0.760	0.052	0.766	0.051
Statlog	**0.786**	0.044	0.758	0.049	0.785	0.045	0.785	0.043	0.779	0.044	0.786	0.043	0.783	0.043
Io	0.950	0.034	0.955	0.034	0.954	0.036	0.950	0.034	0.927	0.048	**0.956**	0.033	0.935	0.037
Wf	0.944	0.009	**0.957**	0.009	0.949	0.010	0.944	0.009	0.957	0.019	0.947	0.008	0.941	0.100
Bio	**0.897**	0.038	0.882	0.037	0.887	0.036	0.894	0.033	0.885	0.040	0.897	0.033	0.891	0.034
Lc	**0.841**	0.032	0.810	0.038	0.802	0.038	0.817	0.037	0.831	0.034	0.839	0.032	0.710	0.045
So	0.836	0.043	0.809	0.047	0.794	0.049	0.825	0.046	0.803	0.047	**0.840**	0.041	0.796	0.048
Average	**0.869**		0.854		0.854		0.860		0.855		0.868		0.837	

In this experiment, all filter ranking methods select out more features than any feature subset selection methods. In the group of feature ranking methods, CCBFS has the fewest number of average selected features.

When looking at the AUC table, CCBFS and ReliefF outperform other filters on the aspect of average AUC. There is no obvious difference between the average AUCs of CCBFS and ReliefF. They both perform well in this test.

While the Vehicle dataset has proved to be a hard task for filters in the test of MLP and Decision Tree, CCBFS in this time improves the AUC comparing with the original dataset.

7.5.4.4 Naïve Bayes Feature Ranking Performance

In this test the AUCs of three feature ranking methods (CCBFS, ReliefF and GainRatio) will be shown by line graphs.

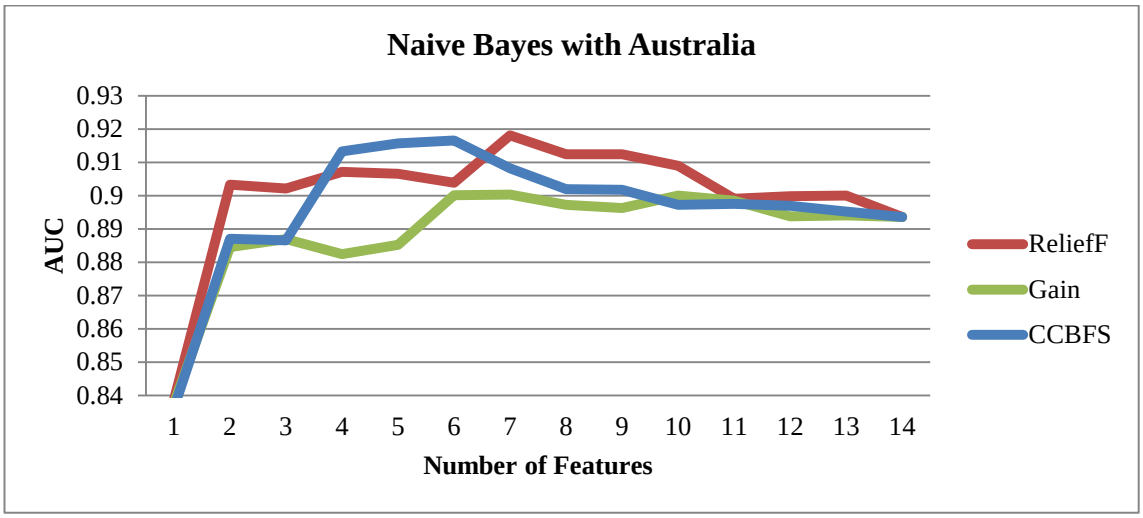

Figure 29 Feature ranking performance under Naïve Bayes with Australia dataset

In this test CCBFS and ReliefF performs better than GainRatio, and both get higher AUCs with smaller feature groups. No obvious declines after the highest point.

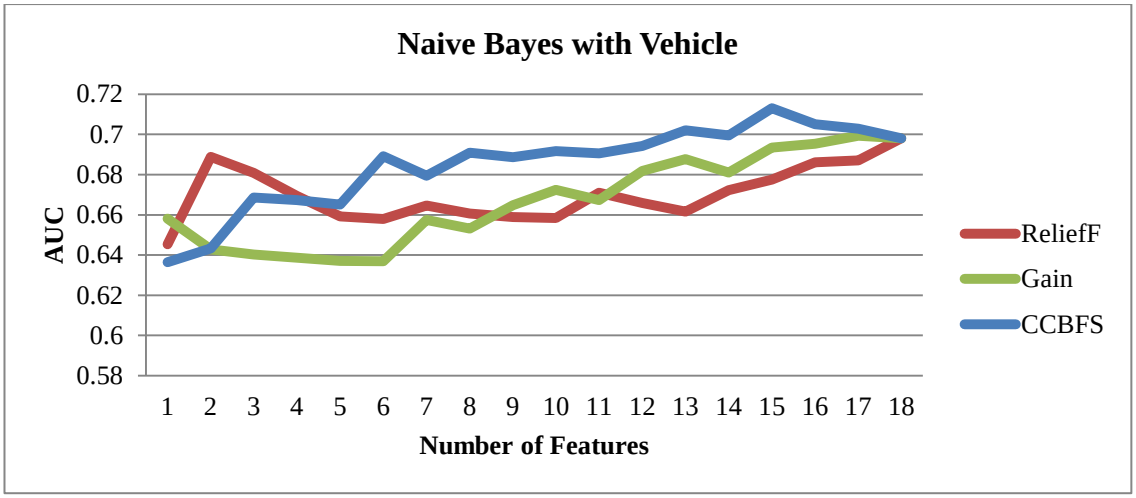

Figure 30 Feature ranking performance under Naïve Bayes with Vehicle dataset

All three lines have raising tendencies in this test. CCBFS has higher AUCs in the end, while ReliefF gets a good feature group in the beginning.

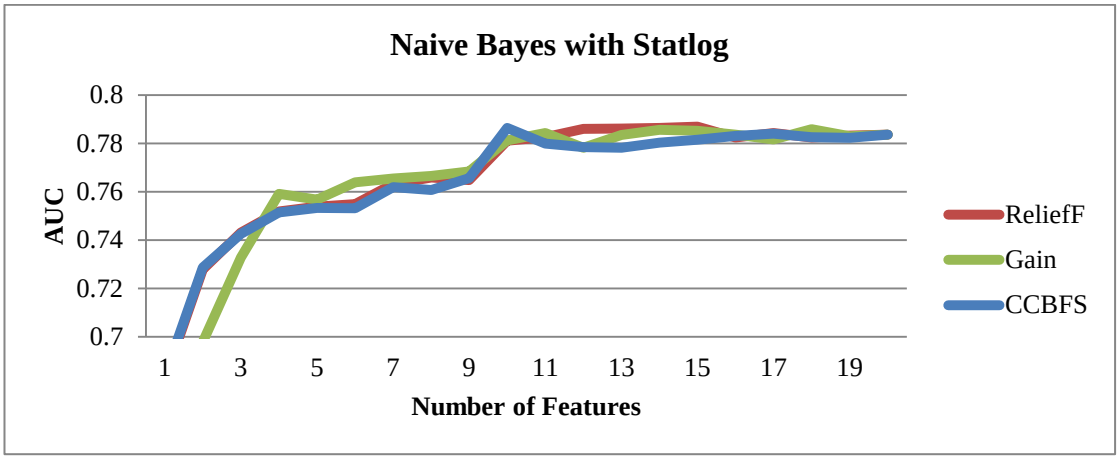

Figure 31 Feature ranking performance under Naïve Bayes with Statlog dataset

All three methods perform similar in the test with Satalog dataset. Actually, no one has good performance as there is no obvious improvement comparing with the original dataset.

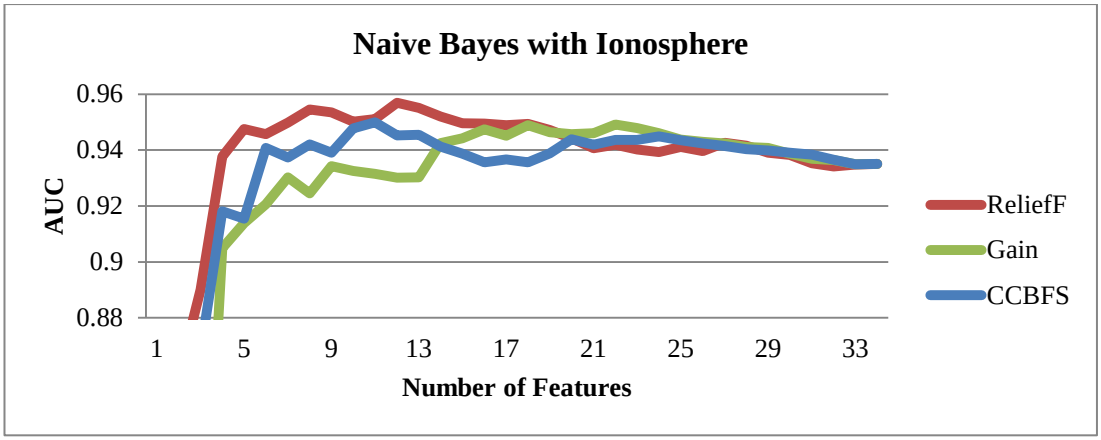

Figure 32 Feature ranking performance under Naïve Bayes with Ionosphere dataset

ReliefF has better performance than the other two methods. All three filters have some improvements on AUC comparing with the original feature group.

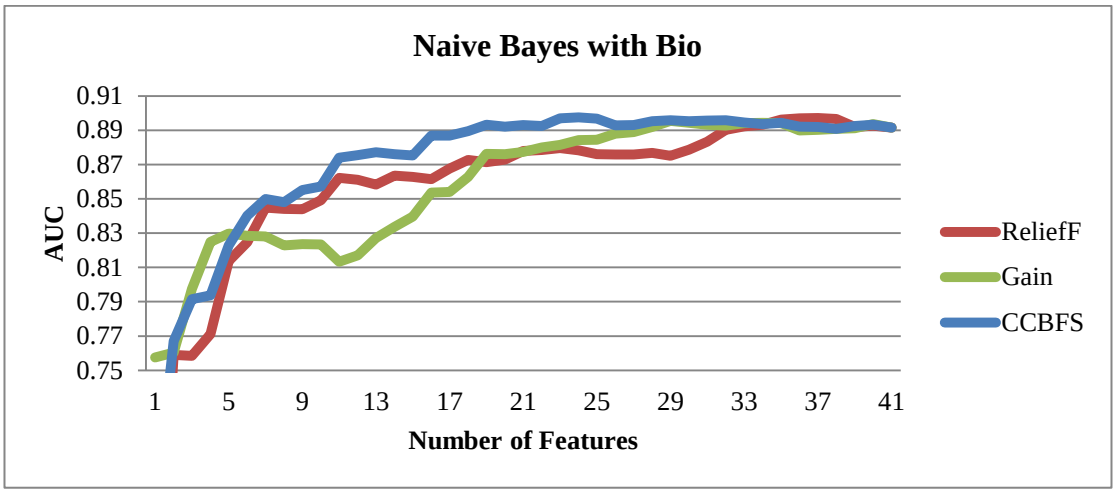

Figure 33 Feature ranking performance under Naïve Bayes with Bio dataset

CCBFS performs slightly better than the other two in the section between 5 and 30 features. Again there is no large improvement achieved by any filters.

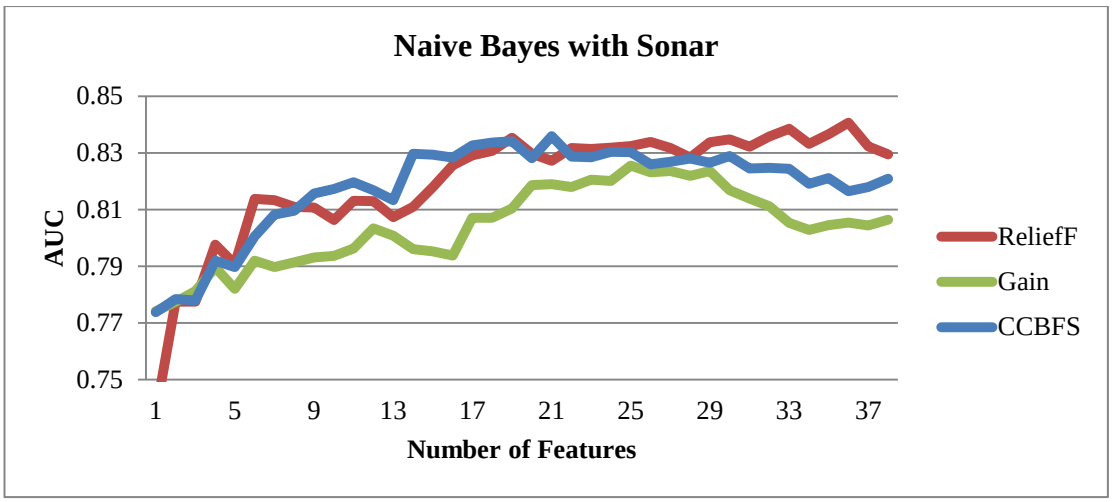

Figure 34 Feature ranking performance under Naïve Bayes with Sonar dataset

CCBFS and ReliefF have higher AUCs than GainRatio when there are more than 5 features selected. ReliefF achieves very high AUC in the end.

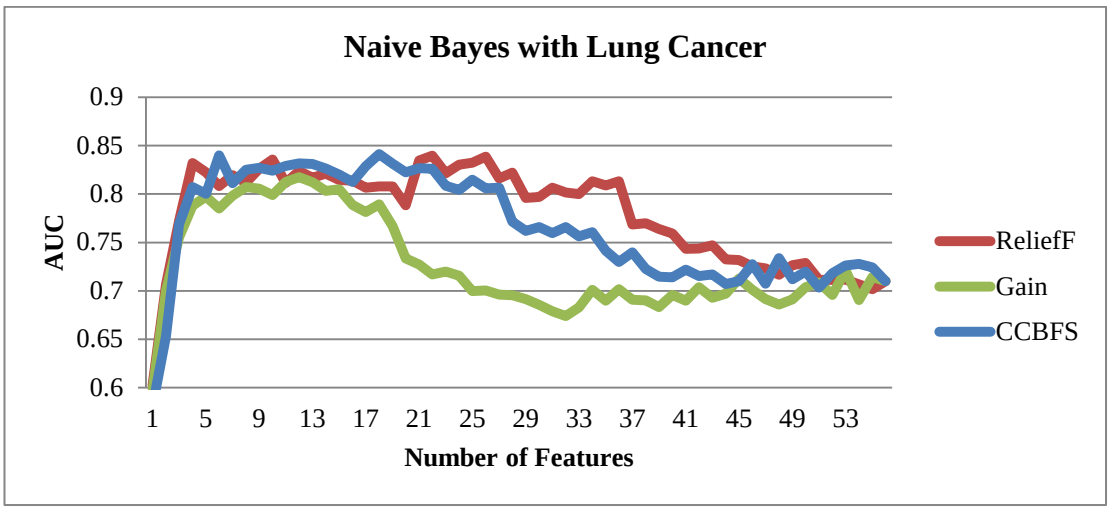

Figure 35 Feature ranking performance under Naïve Bayes with Lung Cancer dataset

Although ReliefF has higher AUCs than CCBFS between 21 and 45 features, CCBFS performs better with small number of features.

7.5.5 Comparison Between Machine Learning Algorithms

In Table 17, the feature selection results by CCBFS and the original dataset are compared between different machine learning algorithms. This is used to find out the relations between feature selection methods and machine learning algorithms.

Table 17 Comparison between algorithms

	MLP CCBFS		MLP Ori		Naïve Bayes CCBFS		Naïve Bayes Ori		J48 CCBFS		J48 Ori	
data	Feature	AUC	Feature	AUC	Feature	AUC	Feature	AUC	Feature	AUC	Feature	AUC
Aus	6	0.92	15	0.914	7	0.917	15	0.875	7	0.898	15	0.862
Ve	19	0.946	19	0.946	16	0.778	19	0.766	10	0.857	19	0.854
Statlog	12	0.742	21	0.726	11	0.786	21	0.783	4	0.715	21	0.647
Io	13	0.943	35	0.903	12	0.95	35	0.935	23	0.915	35	0.891
Wf	19	0.973	41	0.97	14	0.944	41	0.941	11	0.868	41	0.829
Bio	34	0.909	42	0.908	25	0.897	42	0.891	12	0.868	42	0.822
Lc	5	0.799	57	0.662	19	0.841	57	0.71	7	0.75	57	0.586
So	15	0.872	61	0.849	22	0.836	61	0.796	22	0.819	61	0.754
Average	15.375	0.888	36.375	0.86	15.75	0.869	36.375	0.837	12	0.836	36.375	0.781

Firstly, CCBFS improves prediction accuracies for all three machine learning algorithms. Regarding to the average AUCs, CCBFS improves 3.3% for MLP, 3.8% for Naïve Bayes and 7% for J48. CCBFS also selects out fewest features under J48 (12 in average). Thus, compared with MLP and Naïve Bayes, J48 is more sensitive to CCBFS, and CCBFS works better for J48 decision tree among the used dataset.

Secondly, the AUCs of MLP is higher than the other two algorithms. Regarding to the average AUCs for original dataset, MLP is 0.86, higher than Naïve Bayes (0.837) and J48 (0.781). When considering CCBFS results, MLP (0.888) is also higher than Naïve Bayes (0.869) and J48 (0.836). So for these 8 dataset, MLP performs slightly better than the others. However, this is not always true for some specific dataset. When checking the details, MLP has no advantage with some dataset.

At last, it should be aware that CCBFS selects out different feature subsets for different machine learning algorithms. As CCBFS is only a feature ranking method, the subset is given by forward searching according to the ranking list. The differences of CCBFS outputs prove that one feature may be useful for a special machine learning algorithm, but harmful to another algorithm at the same time. That's why wrappers which are specially designed for a pre-selected machine learning algorithm always perform better than filters.

7.5.6 Summary of CCBFS Experiments

Experiment results have proved that CCBFS can improve machine learning performances on many datasets with fewer features. By ranking features according to their consistency and concentration rate, a forward searching according to the feature ranking list can select out small feature groups with high prediction accuracies. Comparisons between other filters show that CCBFS has better performances on all kinds of dataset: dataset with different size and for different real world problem categories. As CCBFS has already been designed to fulfil the special requirements of hybrid feature selection usage, it has all reasons to be chosen as the filter in MCFWFS.

Further experiments prove that CCBFS, as a filter for feature selection, is naturally capable for more than one machine learning algorithms. Except from ANN, it can also improve the performance of Naïve Bayes and Decision Tree as shown in the test results. Comparing with other filters, CCBFS has been proved to be a competitive feature ranking method. It outperforms some famous filters used in this experiment with some dataset. Although CCBFS is designed for ANN hybrid feature selection usage, it is still a good choice of feature selection for other machine learning algorithms.

At last, a comparison between different machine learning algorithms is demonstrated. Sonar dataset is difficult for CCBFS. This is reasonable as there is no prevalent solution for all dataset. Every feature selection algorithm has its strength and limits.

The average AUCs of MLPs with all datasets and feature selection methods are higher than the other two machine learning algorithms. This is a strong prove of the ability of MLP and ANN. It is also the reason that this research chooses ANN as the machine learning method.

7.6 Experiments and Results of Wrapper and Hybrid Feature Selection Method MCFWFS

In this section the experiments of MCFWFS will be designed to show the performance. Comparison between MCFWFS and other feature selection methods will be displayed.

7.6.1 Experiment Setup

As there are many parameters to be set by experiences in GA, AEGA also needs a setup process. Parameter settings of AEGA have been described in section 6.2.

To show the ability of MCFWFS, three ANN models are used in this experiment: MLP, RBF and HONN. MLPs are applied on small, medium and large dataset, with real world problems in categories of classification and recognition. RBF is applied on extremely large dataset, with more than 100 features in dataset. HONN is applied on some small and medium dataset. Some of them are used in CCBFS experiment and others are new ones.

Except AUC which is used to evaluate the prediction accuracy, number of selected features is also used as an important evaluation of feature selection method. Another special evaluation of AEGA is the number of calling fitness function. This is used to show the computing complexity of the GA based searching process: if a searching method can get a good solution with fewer calling of fitness function, then it is thought to be faster and more time consuming. This is especially important for ANN feature selection, because the long training process of ANN is a big problem for wrappers feature selection. This indicator is prevalently used in GA researches.

7.6.2 MCFWFS for MLP

This experiment applies MCFWFS on MLP with all categories of dataset. All dataset have been used in the experiment of CCBFS and introduced in section 7.2.

To demonstrate the performance of MCFWFS, test results are compared with simple GA wrapper and the best filter found in the experiment of CCBFS in section 7.5. SGA is executed in Weka explorer and the setting is as follows:

```
GeneticSearch –Z 20 –G 300 –C 0.6 –M 0.033 –R 20 –S 1
```

This means the maximum generation is 300, crossover rate (probability) is 0.6, mutation rate (probability) is 0.033 and population size is 20. The maximum generation is relatively lower than common settings so the SGA algorithm may highly likely to stop when it reaches the maximum generation number. The reason is that unlike other machine learning algorithms, MLP has longer training time and more complex computing procedure. Although with more generations SGA can achieve better results, the computing time will be extremely long. Even in this experiment, SGA feature selection programs for some datasets can't give out any results in 4 days. These dataset are considered not fit for SGA feature selection.

For each dataset, MCFWFS is applied 10 times and the average and best results are shown in Table 18. Here the best results are those which have highest AUCs. The number of fitness calls represent the running time: more calling of fitness means longer running time.

The results of the best filters are also compared with MCFWFS. In many researches, forward searching with feature ranking list is also considered as a kind of wrapper procedure, as it uses special learning algorithms in the searching process. Because in the experiment of CCBFS, many best results are achieved by forward searching of ranking lists, these results represent comparisons with a large scope of feature selection methods.

Table 18 MCFWFS results with MLP

	MCFWFS Average			MCFWFS Best			SGA			Best Filter		
Data	num fit cal	auc	num feature	num fit cal	auc	num feature	num fit cal	auc	num feature	auc	num feature
Aus	1236	0.925	7.2	1680	0.928	8	30000	0.923	9	0.92	6
Ve	2178.2	0.947	12	3200	0.949	11	30000	0.949	18	0.946	19
Statlog	1242.4	0.749	10	1544	0.75	6	30000	0.752	14	0.742	12
Io	1848	0.965	9.2	1616	0.966	8	30000	0.943	15	0.943	13
Wf	3569.2	0.973	16.4	4640	0.973	15				0.973	16
Bio	1792	0.912	22.8	2106	0.915	25				0.91	33
Lc	3856	0.852	14	3360	0.878	15				0.799	5
So	2563	0.907	18.6	2854	0.915	16				0.876	28

From this table it can be seen that MCFWFS performs much better than filters (including forward searching wrappers) regarding to the AUCs and number of selected features. The average AUCs of all dataset are all not smaller than the best filters' results, while the best AUCs are all larger. There are big differences between the best and

average results, because AEGA is also a kind of GA searching process. It is a nature that the searching results are unstable as GA has no guarantee of finding the “global” optimum. However, considering that there is no precisely definition of the “best” feature group, and the random initialization training of MLP also makes it difficult to consistently evaluate feature groups, finding the global optimum (best feature subset) is unrealistic. This experiment also proves that local optimums have already been better than other feature selection methods. So it has no reason to spend more searching time on finding the global optimum in a feature selection task.

When comparing MCFWFS and SGA under the four dataset with SGA results, MCFWFS has no obvious advantages on AUCs. However, both average results and best results contain fewer features than SGA results. With fewer features, MCFWFS searching results have competitive prediction accuracies. This means there are fewer redundant features in MCFWFS results, and the smaller feature subset can obviously simplify the training process of MLP.

Another thing that should be noticed is the number of fitness calls. All SGA processes stop at the maximum generations. So the callings of fitness are the maximum generations multiply population sizes. As SGA uses the average accuracy of 5 times training (cross validation), the actual number of calling should multiply 5. So all fitness calling numbers are 30000 in this experiment.

The fitness function calling times of MCFWFS are just the population sizes multiply generations. As designed in AEGA, the fitness of each chromosome is the average of AUCs in each generation, and only trains MLP once in one generation. So MCFWFS saves lots of training times, as it need no 5 times average accuracy in each generation. As described in section 6.2.6, some feature groups that are evaluated less than 10 times will be supplied with more evaluations, so there may be more fitness callings in some tests.

The number of fitness callings in MCFWFS is obviously smaller than SGA. This means with less searching time, MCFWFS can find similar (if not better) results than SGA. It should be aware that the best results is a better choice to represent the performance of MCFWFS, as in practical multi times searching is common to be used to find a better solution.

7.6.3 MCFWFS for RBF on extreme large dataset

As dataset are getting larger and larger, 100 features are no longer unusual large for machine learning. Feature selection algorithms will be useless if they can't handle extremely large dataset that has more than 100 features. In this experiment, extremely large dataset are used to test the performance of MCFWFS.

MLP is not fit for machine learning of extremely large dataset. Here another ANN called RBF is used. A brief introduction of RBF is introduced in section 2.2.3. RBF is faster than MLP and still has competitive performances for some datasets. As a kind of ANN, RBF heritages some advantages of ANN and fits for larger dataset.

The setting of MCFWFS is different with the last experiment. A larger population size is needed and along with more generations for searching. The settings are as follows:

Population size (pop_size):200
Maximum generation (max_generation): 600
Crossover rate (pcrossover): 0.6
Mutation rate (pmutate): 0.033
Scale pressure (C): 2
Fitness penalty weight (w): 0.05

Some feature subset selection methods are compared with MCFWFS, including CorrelationFS, ConsistencyFS and INTERACT. Test results are shown in Table 19.

Table 19 MCFWFS for extremely large dataset

	Colon		Lung		Leukemia		Lymphoma	
Data	AUC	Feature	AUC	Feature	AUC	Feature	AUC	Feature
Original	0.652	2001	0.83	326	0.794	7071	0.668	4027
INTERACT	0.903	10	0.809	9	0.945	4	0.924	10
ConsistencyFS	0.853	9	0.774	13	-	-	-	-
CorrelationFS	0.919	23	0.84	72	-	-	-	-
MCFWFS BEST	0.956	22	1	4	1	35	0.993	17
MCFWFS Average	0.943	29.6	1	4.4	0.997	81.4	0.984	19.2

There are no test results in ConsistencyFS and CorrelationFS with dataset Leukemia and Lymphoma, as the programs have no outputs after 3 days running.

Compared with original dataset, MCFWFS results only contains very few features, but with much higher AUCs. As a wrapper process, it has acceptable training time (all tests

stop in 1 day). This is an advantage of MCFWFS comparing with other wrappers as the searching time of wrappers is always too long to be acceptable.

When comparing with other feature subset selection methods, MCFWFS has consistently advantages on the aspect of AUCs. However, MCFWFS may need more features to achieve these AUCs. This is because it uses AUCs in fitness functions, so the searching results are always those with higher AUCs. This is common when comparing wrappers with filters: wrappers can achieve higher accuracies than filters.

Again, the best results of MCFWFS are more representative than average results, as 10 times searching is common in feature selection.

7.6.4 MCFWFS for HONN

To improve the performance of MLP, a new machine learning algorithm called Higher Order Neural Network (HONN) is designed. The basic structure and algorithm is introduced in section 2.2. Although it has higher prediction accuracies than MLP, the disadvantage is also obvious: not all added features are relevant to the target class. Feature selection is essential for HONN on the aspect of simplify feature groups and enhance performances.

In this experiment, the performance of MCFWFS is tested on HONN, comparing with other feature subset selection methods and the original dataset. Only numeric features in dataset are processed into higher order statues. MCFWFSs are applied 10 times to get the best and average results. All AUCs are generated by 10 times validation. All feature selection results are based on HONN.

7.6.4.1 Statlog

In Statlog dataset test, HONN has big improvement on MLP. MCFWFS has obvious advantages regarding to AUCs, and higher AUCs are the main propose of most researches. Although the smallest feature group is achieved by CorrelationFS, the AUC is not satisfied. It is easy to see that there is no relation between number of features and AUCs. Again this proves irrelevant and redundant features are harmful to most machine learning algorithms.

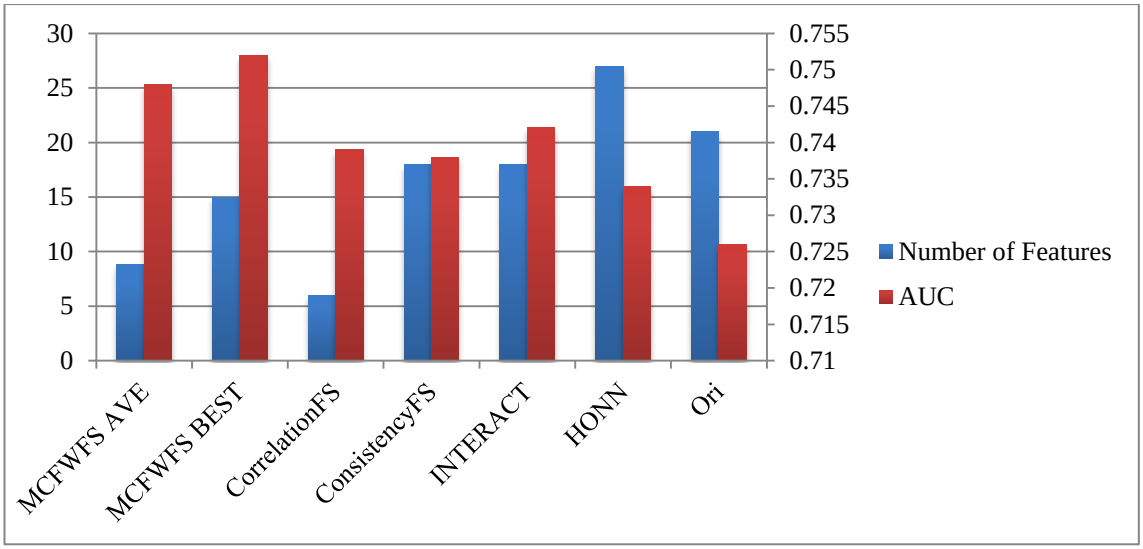

Figure 36 Feature selection results under HONN with Statlog dataset

7.6.4.2 Australia

In the test of Australia credit dataset, MCFWFS's advantages are obvious. The results are stable, with very few features (only more than CorrelationFS) and the highest AUCs. Lots of features are added in HONN but the enhancement of performance is tinny. Other feature selection methods have no effect on improving AUCs.

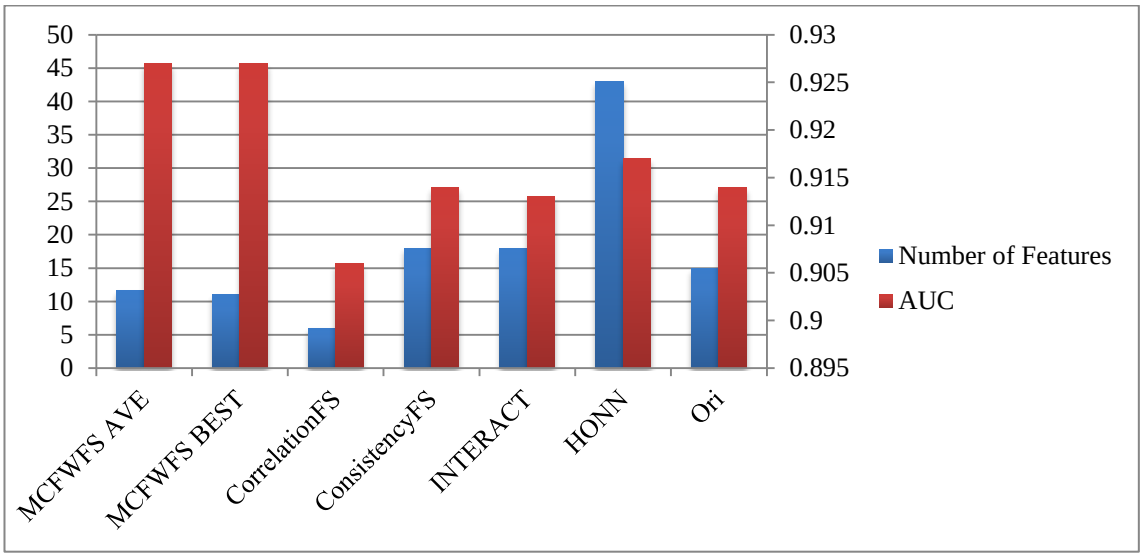

Figure 37 Feature selection results under HONN with Australia dataset

7.6.4.3 Pima

In this test MCFWFS has absolutely advantages comparing with all other groups. With the least number of features, it has the highest AUCs. Other feature selection methods also perform well considering the improved AUCs. A possible explanation is that the added features in HONN contain both useful information and redundancies.

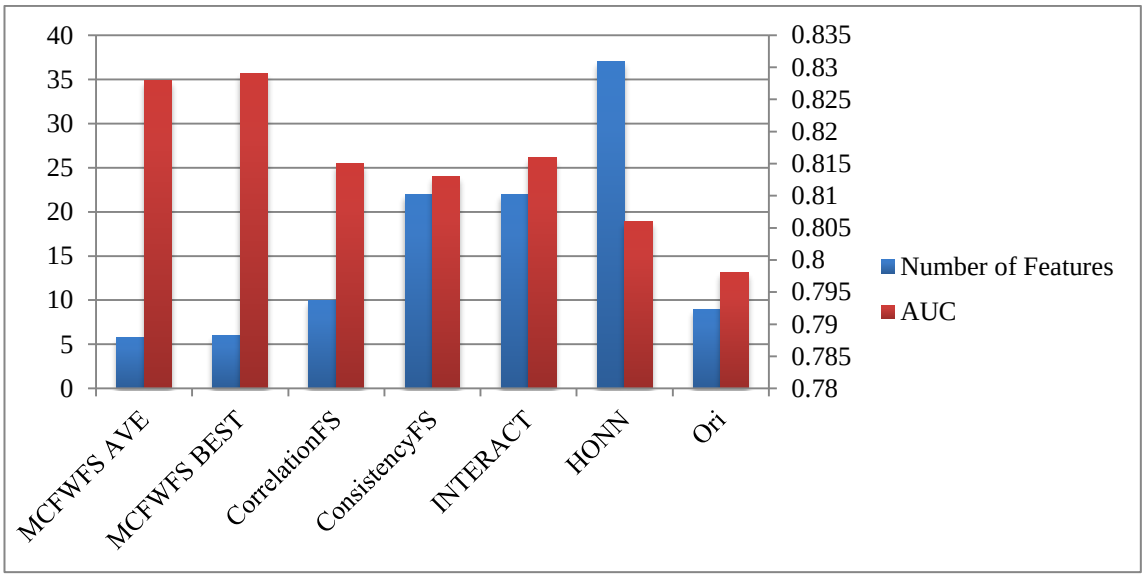

Figure 38 Feature selection results under HONN with Pima dataset

7.6.4.4 Liver

Although MCFWFSs have the highest AUCs, they tend to contain more features than other feature selection methods. This is because of the difference between wrappers and filters. The AUCs of other filters are not acceptable because they are even lower than the original dataset. HONN has limited improvement but when MCFWFS eliminates redundant features, prediction accuracy improves.

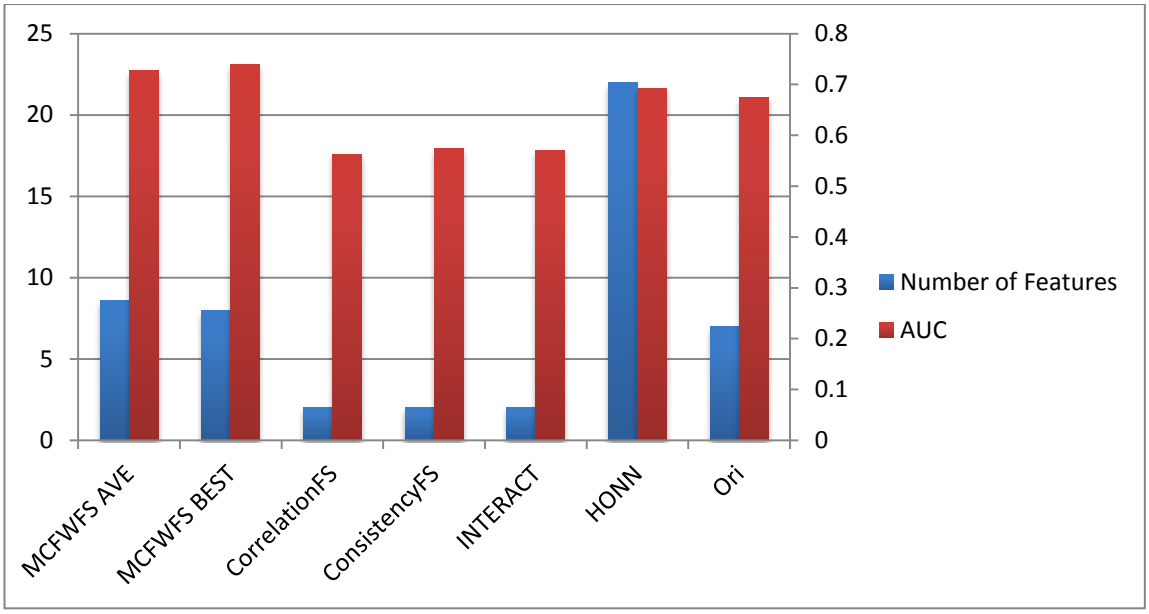

Figure 39 Feature selection results under HONN with Liver dataset

7.6.4.5 Blood

In the test of Blood dataset, all feature selection methods have similar size of feature subsets. However, MCFWFSs have higher accuracies, which means the selected features are more relevant to the target class. This time HONN has huge improvement comparing with original dataset, while filters' performances are bad.

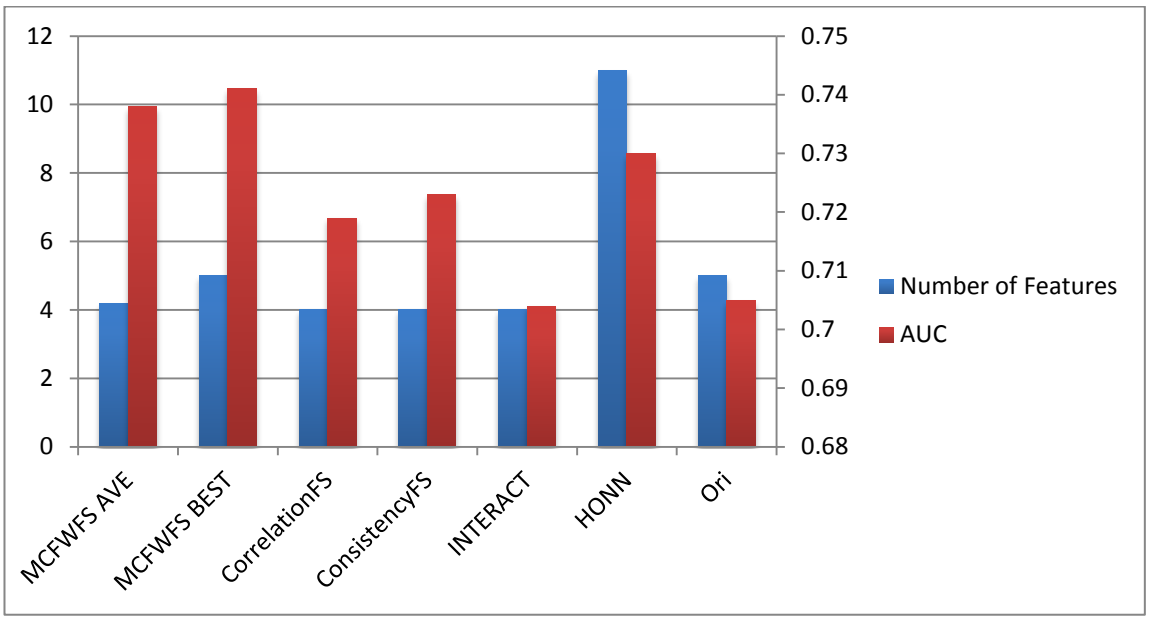

Figure 40 Feature selection results under HONN with Blood dataset

7.6.5 Summary of MCFWFS Experiments

In this experiment three kinds of ANN are used to test the performance of MCFWFS. All these three learning algorithms prove that MCFWFS can enhance the AUC of prediction with small feature subsets.

Comparing with SGA, MCFWFS has competitive results but with much less searching time. For MLP feature selection, this is important because the long training of MLP is a big problem. Dataset with large numbers of features may even unable to apply SGA feature selection on MLP. MCFWFS makes it possible for these dataset achieve better MLP performances with a wrapper process.

Comparing with other feature selection methods, MCFWFS results may contains more features. However, the most important propose of feature selection in this research is enhancing the prediction accuracy. MCFWFS in many tests performs better than other feature selection methods, including both filters and wrappers, feature subset selection algorithms and feature ranking methods, on this point. The comparisons in all experiments prove that MCFWFS has advantages and abilities to improve ANN performance.

The experiment of HONN also provided another way of improving ANN. MCFWFS works well on HONN and in some dataset it is the best choice. The combination of HONN and MCFWFS is the best choice found in the experiments of this research.

7.7 Summary

In this chapter three categories of experimens are demonstrated. All datasets used in experiments are introduced. The concept of AUC explains the reason of chooing it as the evaluation method for this research.

As data preparation is the first step in ANN, experiments of ARCM, a new instance selection method for class imbalance problem, is designed at first. Results support the idea of using ARCM as data preparition method for all tests. The best settings of MLP is also provided in the results, which will be an instruction for the experiments of feature selections.

After that CCBFS is tested with all categories of dataset: small, medium and large, problem of classication and multi class recogniton. Results prove that CCBFS has competitive performances with other filters. A subset of CCBFS ranking list may have higher prediction accuracies than compared feature subset selection algorithms in the results of section 7.5.2. This means CCBFS is suitable to be used as a preselection for hybrid feature selection process. In 7.5.3 CCBFS performs better than the compared feature ranking methods on the aspect of MLP accuracies in a forward searchng process. Thus it is a better choice to be used in local search optimizations for hybrid feature selection. As it is specially designed for the hybrid method, these experiments give all reasons to use CCBFS in the later designed feature selection process.

As an independent filter for feature selection, CCBFS also shows its advantages with other machine learning algorithms such as Decision Tree and Naïve Bayes, under the support of experiment results in section 7.5.4. This provides another choice of feature ranking methods for other researches.

At last, the novel hybrid feature selection method, MCFWFS, is tested under all mentioned dataset. Three kinds of ANN learning methods are tested and MCFWFS shows consistenly good performances on all tests. It has the advantages of fast searching and huge improvement of prediction accuracy. The MCFWFS method can be applied on ANNs for both small and extremely large datasets, with numbers of features vary from 16 to 7021. It can handle both classification problems and multi class recoginition problems with all good performances. It is the most important contribution of this reseach as a fast, prevelant and effecitive feature selection method to improve ANN’s performance.

Chapter 8 - Conclusions

8.1 Introduction

This chapter is a conclusion of the whole study. It starts from a revisiting of literatures, research aims, methodology, designs of novel algorithms and the experiment outcomes. All these works are answers to research questions, and proves of achieving all research aims.

Then the contributions in substantive, methodological and theoretical level are discussed. The main contribution of this research is design of novel algorithms that improve the performance of ANN. These algorithms can be applied on many real world problems and proved to be with good performance.

In the end, advantages and shortages of this study are discussed. Comparing with current methods, novel methods designed in this research have better performances regarding to improve the performance of ANN. However, these algorithms are not perfect and may not fit for all categories of real world problems. Some further work can be done in the future.

8.2 Conclusions

This research starts by a background research of ANN, instance selection and feature selection. ANN is a group of machine learning algorithms that can solve many categories of real world problems. Thus, improving the performance of ANN is with high value, and it is also the main aim of this research.

To achieve this goal, instance selection methods and feature selection methods are investigated. Current instance selection methods that can solve class imbalance problem need to add or delete instances, and this may disturb the later feature selection process. Thus, a novel instance selection method, ARCM, is designed. This novel algorithm can partly solve class imbalance problem for binary classification and enhance ANN prediction accuracy, without add or delete instances.

Besides instance selection, feature selection can also improve the performance of machine learning by removing redundant and irrelevant features and choosing out important features. Filters are fast but wrappers can achieve higher prediction accuracy. While hybrid feature selection methods combines advantages of filters and wrappers, the running time is too long when applied on ANNs. This research designs a novel

hybrid feature selection method by combining a novel filter and a novel wrapper in multiple ways. The new filter CCBFS is based on the concept of consistency in feature selection. It covers the shortages of traditional consistency bases filters, such as no feature ranking, and has more obvious improvement on ANN performances. As current wrappers needs long training time and not fit for ANN, this research designs a novel GA based wrapper AEGA. It solves the slow converging problem and have large searchng scoupe around local optimums. The novel hybrid feature selection algorithm MCFWFS combines CCBFS and AEGA together. It utilizes CCBFS as a preselection step in the initialization of AEGA, and optimize the local searching strategy in AEGA.

All new designs are tested in experiments that cover many categories of real world problems with 15 datasets. Test results show that all these new designs can improve the performance of ANN, with fewer training data (which means shorter training time) and higher prediction accuracies. Comparison tests also indicate that the new algorithms perform better than some current algorithms. It can solve many kinds of real world problems with shorter running time and higher prediction accuracies.

In summary, the contribution of this research can be summarized in the following three levels:

8.2.1 Substantive Level

In substantive level, this research implemented four novel algorithms: ARCM, CCBFS, AEGA and MCFWFS. All these algorithms are realized by Java with Eclipse. Weka machine learning packs are used to help proceed data.

ARCM is a novel instance selection method that can partly solve data imbalance problem. Several credit rating datasets are applied in its experiment. Test results show that ARCM can improve the credit rating accuracy of ANN model. Comparing with some other research results, ARCM has better performance regarding to the classification accuracy. That means ARCM can improve the performance of ANN by partly solving class imbalance problem, and expecially for the credit rating appilcations.

CCBFS is a novel filter for feature selection. It can rank features accordng to their importances, while other consistency based filters can only provide selected feature subsets. Experiments of CCBFS are applied on 8 different datasets, with different kinds

of features (nominal, munerical), different kinds of real world problems (credit rating, disease diagnose, objects recoginization and others), and different size of dataset (small, medium and large dataset). Tests with ANN demonstrate that CCBFS perform better comaring with other filters on most datasets and in average. Feature subset selected by CCBFS has higher ANN prediction accuracies but with fewer features. Tests on Decision Tree (C4.5) and Naive Bayes have similar results. Further comparisons between CCBFS and other feature ranking methods indicate that this novel feature selection algorithm can give those important features with higher scores. Thus, CCBFS is proved to be an important suppliment to consistency based feature selecton algorithms, and is worth to be utilized on many real world problems.

AEGA is an improvement of HGA for feature selection. It is specially designed for MCFWFS hybrid feature selection. Experiments of AEGA and MCFWFS are applied on ANNs such as MLP, RBF and HONN. All these three experiments show that the novel hybrid feature selection algorithm can improve the performance of ANN obviously, from the aspect of higher prediction accuracy and shorter runnings. It makes wrappers possible to be applied on ANN, which used to be too time consuming. Comparing with other feature selection methods, the novel algorithm can quickly give out a proper feature subset, with similar or even higher ANN prediction accuracies. Datasets with different kinds of real world applications have been used, even those with extremely large number of features (more than 7000). Thus, the novel hybrid feature selection algorithm has been proved to be with wide application area and considerable performance.

All these experiments show that this thesis accomplished the task of improving the performance of ANN, and the outcome of this research can be applied on many different categories of real world problems.

8.2.2 Methodological Level

From the point of methodological level, this research utilizes different methodologies for the design of four novel algorithms. The methodology chapter describe these in details.

First, this research analysis the instance selection methods and proposed a novel method to partly solve class imbalance problem without adding or delete instances. This new method is based on resamplying methods, and aiming at improving the machine learning performances.

Then the systematic study of filters is presented and proposed a novel filter based on the concept of consistency in feature selection. This novel consistency based filter utilizes the defination of consistency and inconsistency in other researches, but with little modification. The methodology study of feature ranking methods are used to design a novel kind of solutions: consistency based feature ranking filters.

The study of wrappers and hybrid feature selection methods forms the design of AEGA and MCFWFS. GA is chosen as the searching strategy of wrapper in this reseach and the elitism methods are added into GA. Studies of hybrid feature selection methods indicate that there are mutiple ways of combining filters with wrappers, and for diffirient proposes. Utilizing filters before wrappers can be seen as a pre selection step that decrease running time, and using filters in GA based wrappers can optimize converging process. Some new designs are also added into these methods, such as novel fitness function, novel elitist list and others.

8.2.3 Theoretical Level

This research focuses on solving real world problems. To achieve the main aim that is improving the performance of ANN, designs of novel algorithms and the realization of them are main contributions.

As to theoretical level contributions, this study proposes novel definations of Consistency-Concentration Rate and the calculation of it. It is a supplyment to the theory of consistency in feature selection. This new theory makes consistency based filters able to provide feature ranking lists, and at the same time takes “concentration” of each feature into consideration.

The design of AEGA is also a supplyment to GA based wrapper theory. It brings the theory of elitism GA into the feature selection areas.

8.3 Research Questions Solved

As discussed in section 1.4, there are four questions to be answered in this research. So in the first and very important part of this conclusion, solutions will be demonstrated to prove that each aim of this research has been achieved.

1. Design data preparing method

This thesis has shown that the novel instance selection method, ARCM, has good performance on solving class imbalance problem when it is used in data preparing. It chooses instances for training, validation and test by random, and at the same time keeps the ratio of each class equal in each instance groups. Experiment results have proved that this data preparing method can improve the classification accuracy of ANN. It can be another choice of data pre-processing method to solve class imbalance problem.

2. Design filter

The novel filter designed in this thesis is called CCBFS and described in chapter 5. To fulfil the requirements of hybrid feature selection, CCBFS is a feature ranking method that evaluates each feature independently. Experiments of CCBFS are well designed for both ANN and other machine learning algorithms. Results have shown that CCBFS with forward searching process can generate a small feature subset with higher prediction accuracies, comparing with the original dataset and some well-known feature subset selection methods. More objective comparisons between CCBFS and other feature ranking methods show the advantages of CCBFS clearly. Supplementary tests by Decision Tree and Naïve Bayes also provide strong evidence that CCBFS is a competitive feature selection method. It can not only be used in the hybrid feature selection method of this research, but also a good choice for other work.

3. Design wrapper

In this research a novel wrapper called AEGA is designed on the base of Genetic Algorithm. This searching process is specially designed for ANN: comparing with simple GA, AEGA cuts down the number of fitness calling, but generates competitive results. The performance of AEGA can only be shown in experiments of hybrid feature

selection methods, as this wrapper need the support of local optimization which is a combination of filters.

4. Design hybrid feature selection

The novel hybrid feature selection method, MCFWFS, combines filters and wrappers in two ways to simplify searching process. Experiments of MCFWFS demonstrate that comparing with SGA, it needs much fewer calling of fitness functions, which means less computing time, to output a competitive solution. Tests of MCFWFS have been done on three kinds of ANN: MLP, RBF and HONN. In all three tests CCBFS improves ANN performance significantly. It makes it possible to apply a wrapper feature selection process on extremely large dataset under ANN, which was thought to be too time consuming to execute. MCFWFS has been proved to be suitable for ANN feature selection on many kinds of datasets, with high computing speed and great improvement of ANN performances.

8.4 Advantages

Advantages of the new hybrid feature selection algorithm MCFWFS includes fast searching speed and large improvements for ANN. To be more specific, it has the following advantages:

(1) It can improve the performance of many kinds of ANN, including MLP, RBF and HONN. Tests of all these three ANNs demonstrated significant improvements of prediction accuracies and reduce of feature numbers.

(2) It can handle with prevalent real world problems in categories of both classification and multi class recognition. These are two prevalent kinds of problems, including credit card fraud detection, medical diagnose, weather prediction and object recognition.

(3) It has fast selection speed comparing with SGA wrappers. It needs fewer calling of fitness function, which is the most time consuming step of ANN wrappers. The 10 times average fitness value also provides similar stability as SGA and other wrappers.

(4) It can handle with all size of dataset, from small dataset with less than 20 features, to extremely large dataset with more than 7000 features.

(5) It needs relatively simple parameter settings, comparing with other hybrid feature selection methods. The adaptive initialization method simplifies a lot of work. Only the size of population and maximum generation need to be changed when applying on different dataset.

(6) The filter process, CCBFS, can also be used individually in other works. It has good performance in the filter tests. With a forward searching, it can generate small feature groups with high prediction accuracies. Comparing with other feature ranking methods, its outputs have better performances with the same amounts of features.

(7) The data preparing method ARCM can also be used individually in other work to solve class imbalance problem. Experiment results show that it is effective on improving classification accuracies.

8.5 Limitations

While there are many advantages of MCFWFS, there are some limitations exist

(1) MCFWFS cannot process regression problems. There are two reasons: first, CCBFS can only process classification and recognition problems; second, the fitness function in MCFWFS cannot handle with regression problems.

(2) The outputs of MCFWFS are unstable. This is one of the characters of genetic algorithm, as the searching process may fall into local optimum but not always the global optimum.

(3) Missing values in dataset are not considered in MCFWFS. It only treat the missing values as a special value in dataset.

8.6 Further Work

Here are some further work about both MCFWFS and the topic of this thesis.

(1) To apply MCFWFS on more kinds of dataset, changes of CCBFS and fitness functions are necessary to adapt regression problems. The change of CCBFS may exist on altering the calculation method of consistency correlation rate. The fitness function can use measurements other than classification accuracies to represent regression accuracies. This work has high value as regression problems are common in real world.

(2) Support Vector Machine (SVM) is also a very powerful machine learning algorithm and has been proved to be efficient. Altering MCFWFS to adapt to SVM feature selection will be worth researching.

(3) Dataset with extremely imbalance classes are difficult to apply feature selection methods, and cannot be solved by ARCM. However, this kind of dataset really exists in real world, such as diagnose of cancer, prediction of special events and so on. Also, ARCM can only handle dataset with two classes. How to handle extremely imbalance dataset and more categories of real world datasets will be worth researching.

8.7 Summary

In this chapter the conclusion of the whole work is discussed, along with the possible further work. This research has achieved the main aim: improving the performance of ANN. Apart from this main contribution, this study also has contributions of substantive, methodological and theoretical level. It answered all research questions that are proposed in the beginning, and achieved all corresponding aims.

The solutions provided in this study has many advantages and has been proved to have good performances on solving real world problems. It can be considered as a competitive supplyment to feature selection methods for ANN. However, there are still limitations and these should be noticed and solved in the future.

Chapter 9 - Bibliography

ALBA, E., GARCIA-NIETO, J., JOURDAN, L. & TALBI, E. G. Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms. 2007 IEEE Congress on Evolutionary Computation, 25-28 Sept. 2007 2007. 284-290.

ALMUALLIM, H. & DIETTERICH, T. G. Learning with many irrelevant features. AAAI, 1991. Citeseer, 547-552.

ALMUALLIM, H. & DIETTERICH, T. G. 1994. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence,* 69**,** 279-305.

ARAUZO, A., BENITEZ, J. M. & CASTRO, J. L. 2003. C-focus: A continuous extension of focus. *Advances in Soft Computing.* Springer.

BACCIANELLA, S., ESULI, A. & SEBASTIANI, F. 2014. Feature selection for ordinal text classification. *Neural Computation,* 26**,** 557-591.

BERMEJO, P., DE LA OSSA, L., GAMEZ, J. A. & PUERTA, J. M. 2012. Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking. *Knowledge-Based Systems,* 25**,** 35-44.

BHOWAN, U., JOHNSTON, M., MENGJIE, Z. & XIN, Y. 2013. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *Evolutionary Computation, IEEE Transactions on,* 17**,** 368-386.

BLUM, A. L. & LANGLEY, P. 1997. Relevance selection of relevant features and examples in machine learning. *Artificial Intelligence,* 97**,** 245-271.

BOL N-CANEDO, V., PORTO-D AZ, I., S NCHEZ-MARO O, N. & ALONSO-BETANZOS, A. 2014. A framework for cost-based feature selection. *Pattern Recognition*.

BOROS, E., HAMMER, P. L., IBARAKI, T., KOGAN, A., MAYORAZ, E. & MUCHNIK, I. 2000. An implementation of logical analysis of data. *Knowledge and Data Engineering, IEEE Transactions on,* 12**,** 292-306.

BRILL, F. Z., BROWN, D. E. & MARTIN, W. N. 1992. Fast generic selection of features for neural network classifiers. *Neural Networks, IEEE Transactions on,* 3**,** 324-328.

BROOMHEAD, D. S. & LOWE, D. 1988. Radial basis functions, multi-variable functional interpolation and adaptive networks. DTIC Document.

BROWN, I. & MUES, C. 2012. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications,* 39**,** 3446-3453.

BUI, T. N. & MOON, B. R. 1996. Genetic algorithm and graph partitioning. *Computers, IEEE Transactions on,* 45**,** 841-855.

CADENAS, J. M., GARRIDO, M. C. & MART NEZ, R. 2013. Feature subset selection Filter–Wrapper based on low quality data. *Expert Systems with Applications,* 40**,** 6241-6252.

CANT -PAZ, E. 1998. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis,* 10**,** 141-171.

CASTRO, C. L. & BRAGA, A. P. 2013. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *Neural Networks and Learning Systems, IEEE Transactions on,* 24**,** 888-899.

CHAKRABORTY, B. & CHAKRABORTY, G. Fuzzy consistency measure with particle swarm optimization for feature selection. 2013 IEEE International Conference on Systems, Man, and Cybernetics, 13-16 Oct. 2013. 4311-4315.

CHAKRABORTY, R. & PAL, N. R. 2014. Feature selection using a neural framework with controlled redundancy. *Neural Networks and Learning Systems, IEEE Transactions on,* PP**,** 1-1.

CHANDRASHEKAR, G. & SAHIN, F. 2014. A survey on feature selection methods. *Computers & Electrical Engineering,* 40**,** 16-28.

CHAPELLE, O. & KEERTHI, S. S. Multi-class feature selection with support vector machines. Proceedings of the American statistical association, 2008.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O. & KEGELMEYER, W. P. 2011. SMOTE: synthetic minority over-sampling technique. *arXiv preprint arXiv:1106.1813*.

CHEN, X.-W. & WASIKOWSKI, M. 2008. FAST: a roc-based feature selection metric for small samples and imbalanced data classification problems. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* Las Vegas, Nevada, USA: ACM.

CINAR, E. & SAHIN, F. A study of recent classification algorithms and a novel approach for EEG data classification. Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, 2010. IEEE, 3366-3372.

CORD N, O., DAMAS, S. & SANTAMAR A, J. 2006. Feature-based image registration by means of the CHC evolutionary algorithm. *Image and Vision Computing,* 24**,** 525-533.

DAS, S. Filters, wrappers and a boosting-based hybrid for feature selection. ICML, 2001. Citeseer, 74-81.

DASH, M. & LIU, H. 1997. Feature selection for classification. *Intelligent Data Analysis,* 1**,** 131-156.

DASH, M., LIU, H. & MOTODA, H. 2000. Consistency based feature selection. *Knowledge Discovery and Data Mining. Current Issues and New Applications.* Springer.

DAVIS, L. 1991. *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, pp.23-45.

DAYHOFF, J. E. 1996. *Neural Network Architectures: An Introduction*, International Thomson Computer Press. pp. 1-36.

DING, C. & PENG, H. 2005. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology,* 03**,** 185-205.

ESPEJO, P. G., VENTURA, S. & HERRERA, F. 2010. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* 40**,** 121-144.

ESTEVEZ, P. A., TESMER, M., PEREZ, C. A. & ZURADA, J. M. 2009. Normalized Mutual Information Feature Selection. *Neural Networks, IEEE Transactions on,* 20**,** 189-201.

FAWCETT, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters,* 27**,** 861-874.

FOGARTY, J., BAKER, R. S. & HUDSON, S. E. 2005. Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction. *Proceedings of Graphics Interface 2005.* Victoria, British Columbia: Canadian Human-Computer Communications Society.

FOROUTAN, I. & SKLANSKY, J. 1985. Feature selection for automatic classification of non-gaussian data. *IEEE Transactions on Systems, Man and Cybernetics,* SMC-17**,** 187-198.

FULCHER, J., ZHANG, M. & XU, S. 2006. Application of higher-order neural networks to financial time-series prediction. *Artificial neural networks in finance and manufacturing***,** 80-108.

GALAR, M., FERNA, X, NDEZ, A., BARRENECHEA, E., BUSTINCE, H. & HERRERA, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* 42**,** 463-484.

GARC, A-PEDRAJAS, N., DE, H.-G., A, A., REZ, R. & GUEZ, J. 2014. A scalable memetic algorithm for simultaneous instance and feature selection. *Evolutionary Computation,* 22**,** 1-45.

GARCI, X, A-PEDRAJAS, N., PEREZ, R., X, GUEZ, J., DE, H.-G., X & A, A. 2013. OligoIS: scalable instance selection for class-imbalanced data sets. *Cybernetics, IEEE Transactions on,* 43**,** 332-346.

GILES, C. L. & MAXWELL, T. 1987. Learning, invariance, and generalization in high-order neural networks. *Applied Optics,* 26**,** 4972-4978.

GOLUB, T. R., SLONIM, D. K., TAMAYO, P., HUARD, C., GAASENBEEK, M., MESIROV, J. P., COLLER, H., LOH, M. L., DOWNING, J. R. & CALIGIURI, M. A. 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science,* 286**,** 531-537.

GONEN, G. B., GONEN, M. & GURGEN, F. 2012. Probabilistic and discriminative group-wise feature selection methods for credit risk analysis. *Expert Systems with Applications,* 39**,** 11709-11717.

GUTLEIN, M., FRANK, E., HALL, M. & KARWATH, A. Large-scale attribute selection using wrappers. Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on, 2009. IEEE, 332-339.

GUYON, I. & ELISSEEFF, A. 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research,* 3**,** 1157-1182.

GUYON, I., WESTON, J., BARNHILL, S. & VAPNIK, V. 2002. Gene selection for cancer classification using support vector machines. *Machine Learning,* 46**,** 389-422.

HAIBO, H., YANG, B., GARCIA, E. A. & SHUTAO, L. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, 1-8 June 2008 2008. 1322-1328.

HALL, M. A. 1999. *Correlation-based feature selection for machine learning.* The University of Waikato.

HALL, M. A. & HOLMES, G. 2003. Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on,* 15**,** 1437-1447.

HAN, H., WANG, W.-Y. & MAO, B.-H. 2005. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. *Advances in Intelligent Computing.* Springer.

HANCHUAN, P., FUHUI, L. & DING, C. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 27**,** 1226-1238.

HAYKIN, S. 1998. *Neural Networks: A Comprehensive Foundation,* Upper Saddle River, N.J. : Prentice Hall PTR. pp. 178-277.

HE, H. & GARCIA, E. A. 2009. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on,* 21**,** 1263-1284.

HE, J. & LIU, H. The application of dynamic k-means clustering algorithm in the center selection of RBF neural networks. Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, 14-17 Oct. 2009 2009. 488-491.

HEBB, D. O. 2005. *The organization of behavior: A neuropsychological theory,* Psychology Press. pp. 60-78.

HENS, A. B. & TIWARI, M. K. 2012. Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. *Expert Systems with Applications,* 39**,** 6774-6781.

HINTON, G., DENG, L., YU, D., DAHL, G. E., MOHAMED, A.-R., JAITLY, N., SENIOR, A., VANHOUCKE, V., NGUYEN, P. & SAINATH, T. N. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE,* 29**,** 82-97.

HINTON, G. E., OSINDERO, S. & TEH, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural computation,* 18**,** 1527-1554.

HOLLAND, J. H. 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence,* U Michigan Press.

HONG-GUI, H., LI-DAN, W. & JUN-FEI, Q. 2013. Efficient self-organizing multilayer neural network for nonlinear system modeling. *Neural Networks,* 43**,** 22-32.

HUANG, J., CAI, Y. & XU, X. A wrapper for feature selection based on mutual information. Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 2006. IEEE, 618-621.

JALALI, L., NASIRI, M. & MINAEI, B. A hybrid feature selection method based on fuzzy feature selection and consistency measures. 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, 20-22 Nov. 2009. 718-722.

JAPKOWICZ, N. Learning from imbalanced data sets: a comparison of various strategies. AAAI workshop on learning from imbalanced data sets, 2000. Menlo Park, CA.

JIYE, L., FENG, W., CHUANGYIN, D. & YUHUA, Q. 2014. A group incremental approach to feature selection applying rough set technique. *Knowledge and Data Engineering, IEEE Transactions on,* 26**,** 294-308.

JOG, P., SUH, J. Y. & GUCHT, D. V. 1989. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. *Proceedings of the third international conference on Genetic algorithms.* George Mason University, USA: Morgan Kaufmann Publishers Inc.

KABIR, M. M., SHAHJAHAN, M. & MURASE, K. 2011. A new local search based hybrid genetic algorithm for feature selection. *Neurocomputing,* 74**,** 2914-2928.

KARABULUT, E. M., ÖZEL, S. A. & İBRIK I, T. 2012. A comparative study on the effect of feature selection on classification accuracy. *Procedia Technology,* 1**,** 323-327.

KARKKAINEN, T. & HEIKKOLA, E. 2004. Robust formulations for training multilayer perceptrons. *Neural Computation,* 16**,** 837-862.

KERSTEN, J. 2014. Simultaneous feature selection and Gaussian mixture model estimation for supervised classification problems. *Pattern Recognition*.

KHASHEI, M., REZVAN, M. T., HAMADANI, A. Z. & BIJARI, M. 2013. A bi-level neural-based fuzzy classification approach for credit scoring problems. *Complexity,* 18**,** 46-57.

KHASHMAN, A. 2011. Credit risk evaluation using neural networks: Emotional versus conventional models. *Applied Soft Computing,* 11**,** 5477-5484.

KHOSHGOFTAAR, T. M., VAN HULSE, J. & NAPOLITANO, A. 2010. Supervised neural network modeling: an empirical investigation into learning from imbalanced data with labeling errors. *Neural Networks, IEEE Transactions on,* 21**,** 813-830.

KHOSHGOFTAAR, T. M., VAN HULSE, J. & NAPOLITANO, A. 2011. Comparing boosting and bagging techniques with noisy and imbalanced data. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* 41**,** 552-568.

KIRA, K. & RENDELL, L. A. A practical approach to feature selection. Proceedings of the ninth international workshop on machine learning, 1992. 249-256.

KOHAVI, R. & JOHN, G. H. 1997. Wrappers for feature subset selection. *Artificial Intelligence,* 97**,** 273-324.

KONONENKO, I. Estimating attributes: analysis and extensions of RELIEF. Machine Learning: ECML-94, 1994. Springer, 171-182.

KOSMATOPOULOS, E. B., POLYCARPOU, M. M., CHRISTODOULOU, M. A. & IOANNOU, P. A. 1995. High-order neural network structures for identification of dynamical systems. *Neural Networks, IEEE Transactions on,* 6**,** 422-431.

KOTSIANTIS, S. B., ZAHARAKIS, I. & PINTELAS, P. 2007. Supervised machine learning: A review of classification techniques. *Informatica,* 31**,** 249-268.

KUDO, M. & SKLANSKY, J. 2000. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition,* 33**,** 25-41.

KUNCHEVA, L. I. 1992. Fuzzy rough sets: application to feature selection. *Fuzzy sets and Systems,* 51**,** 147-153.

KUNCHEVA, L. I. & JAIN, L. C. 1999. Nearest neighbor classifier: simultaneous editing and feature selection. *Pattern Recognition Letters,* 20**,** 1149-1156.

KWAK, N. & CHONG-HO, C. 2002. Input feature selection for classification problems. *Neural Networks, IEEE Transactions on,* 13**,** 143-159.

LANGLEY, P. & SIMON, H. A. 1995. Applications of machine learning and rule induction. *Communications of the ACM,* 38**,** 54-64.

LI, J., CHENG, K., WANG, S., MORSTATTER, F., TREVINO, R. P., TANG, J. & LIU, H. 2016. Feature selection: a data perspective. *arXiv preprint arXiv:1601.07996*.

LI, W., WANG, Y., LI, W., ZHANG, J. & JINYAN, L. Sparselized higher-order neural network and its pruning algorithm. Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on, 4-8 May 1998 1998. 359-362 vol.1.

LICHMAN, M. 2013. *{UCI} Machine Learning Repository* [Online]. University of California, Irvine, School of Information and Computer Sciences. Available: http://archive.ics.uci.edu/ml [Accessed].

LIU, H. 2010. *Instance Selection and Construction for Data Mining*, Springer-Verlag. pp. 3-20.

LIU, H., MOTODA, H. & DASH, M. 1998. A monotonic measure for optimal feature selection. *Machine Learning: ECML-98*. Springer.

LIU, H. & YU, L. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17**,** 491-502.

LIU, S., MCGREE, J., GE, Z. & XIE, Y. 2015. *Computational and Statistical Methods for Analysing Big Data with Applications*, Academic Press. pp. 1-28.

LOONG, A. S., CHOON, O. H. & CHIN, L. H. 2008. Criterion in selecting the clustering algorithm in radial basis functional link nets. *WSEAS Transactions on Systems*, 7**,** 1290-1299.

LUPING, Z., LEI, W. & CHUNHUA, S. 2010. Feature selection with redundancy-constrained class separability. *Neural Networks, IEEE Transactions on*, 21**,** 853-858.

MAHALAKSHMI, S. & SIVASANKAR, E. Cross domain sentiment analysis using different machine learning techniques. Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO-2015), 2015. Springer, 77-87.

MARCANO-CEDE O, A., MARIN-DE-LA-BARCENA, A., JIMENEZ-TRILLO, J., PI UELA, J. A. & ANDINA, D. 2011. Artificial metaplasticity neural network applied to credit scoring. *International Journal of Neural Systems*, 21**,** 311-317.

MARCOS, J. V., HORNERO, R., ÁLVAREZ, D., DEL CAMPO, F., L PEZ, M. & ZAMARR N, C. 2007. Radial basis function classifiers to help in the diagnosis of the obstructive sleep apnoea syndrome from nocturnal oximetry. *Medical & Biological Engineering & Computing*, 46**,** 323-332.

MARCZYK, A. 2004. *Genetic Algorithms and Evolutionary Computation* [Online]. TalkOrigins: TalkOrigins. Available: http://www.talkorigins.org/faqs/genalg/genalg.html [Accessed 2004].

MARQU S, A. I., GARC A, V. & S NCHEZ, J. S. 2013. On the suitability of resampling techniques for the class imbalance problem in credit scoring. *Journal of the Operational Research Society*, 64**,** 1060-1070.

MCCULLOCH, W. S. & PITTS, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5**,** 115-133.

MINGKUI, T., TSANG, I. W. & LI, W. 2013. Minimax sparse logistic regression for very high-dimensional feature selection. *Neural Networks and Learning Systems, IEEE Transactions on*, 24**,** 1609-1622.

MINLONG, L., KE, T. & XIN, Y. 2013. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *Neural Networks and Learning Systems, IEEE Transactions on*, 24**,** 647-660.

MUNDRA, P. A. & RAJAPAKSE, J. C. 2010. SVM-RFE with MRMR filter for gene selection. *IEEE transactions on nanobioscience*, 9**,** 31-37.

NAKARIYAKUL, S. & CASASENT, D. P. 2009. An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42**,** 1932-1940.

OH, I.-S., LEE, J.-S. & MOON, B.-R. 2004. Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26**,** 1424-1437.

OLIVEIRA, L. S., SABOURIN, R., BORTOLOZZI, F. & SUEN, C. Y. 2003. A methodology for feature selection using multiobjective genetic algorithms for

handwritten digit string recognition. *International Journal of Pattern Recognition and Artificial Intelligence,* 17**,** 903-929.

PARTHALAIN, N., QIANG, S. & JENSEN, R. 2010. A distance measure approach to exploring the rough set boundary region for attribute reduction. *Knowledge and Data Engineering, IEEE Transactions on,* 22**,** 305-317.

PAWLAK, Z. 1982. Rough sets. *International Journal of Computer & Information Sciences,* 11**,** 341-356.

PAWLAK, Z. 1991. *Rough sets: Theoretical aspects of reasoning about data*, Springer. pp. 9-32.

PING, Y. & YONGHENG, L. 2011. Neighborhood rough set and SVM based hybrid credit scoring classifier. *Expert Systems with Applications,* 38**,** 11300-11304.

PUDIL, P., NOVOVIČOV , J. & KITTLER, J. 1994. Floating search methods in feature selection. *Pattern Recognition Letters,* 15**,** 1119-1125.

QUINLAN, J. R. 1993. *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc. pp. 4-11.

RAMONA, M., RICHARD, G. & DAVID, B. 2012. Multiclass feature selection with kernel gram-matrix-based criteria. *Neural Networks and Learning Systems, IEEE Transactions on,* 23**,** 1611-1623.

RAYMER, M. L., PUNCH, W. F., GOODMAN, E. D., KUHN, L. A. & JAIN, A. K. 2000. Dimensionality reduction using genetic algorithms. *Evolutionary Computation, IEEE Transactions on,* 4**,** 164-171.

REDDING, N. J., KOWALCZYK, A. & DOWNS, T. 1993. Constructive higher-order network that is polynomial time. *Neural Networks,* 6**,** 997-1010.

ROMERO, E. & SOPENA, J. M. 2008. Performing feature selection with multilayer perceptrons. *IEEE Transactions on Neural Networks,* 19**,** 431-441.

ROSENBLATT, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review,* 65**,** 386.

RUIZ, R., RIQUELME, J. C. & AGUILAR-RUIZ, J. S. 2008. Best agglomerative ranked subset for feature selection. *Journal of Machine Learning Research-Proceedings Track,* 4**,** 148-162.

SADJADI, F. Comparison of fitness scaling functions in genetic algorithms with applications to optical processing. Optical Science and Technology, the SPIE 49th Annual Meeting, 2004. International Society for Optics and Photonics, 356-364.

SARAFRAZI, S. & NEZAMABADI-POUR, H. 2013. Facing the classification of binary problems with a GSA-SVM hybrid system. *Mathematical and Computer Modelling,* 57**,** 270-278.

SCHMIDT, W. A. C. & DAVIS, J. P. 1993. Pattern recognition properties of various feature spaces for higher order neural networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 15**,** 795-801.

SETIONO, R., BAESENS, B. & MUES, C. 2011. Rule extraction from minimal neural networks for credit card screening. *International Journal of Neural Systems,* 21**,** 265-276.

SETIONO, R. & HUAN, L. 1997. Neural-network feature selector. *Neural Networks, IEEE Transactions on,* 8**,** 654-662.

SETIONO, R. & LIU, H. 1997. Neural-network feature selector. *IEEE Transactions on Neural Networks,* 8**,** 654-662.

SHAHZAD, S. I. A. W. 2012. A feature subset selection method based on symmetric uncertainty and ant colony optimization. *International Journal of Computer Applications,* 60**,** 5-10.

SHENG, W. & BANTA, L. E. 2006. Parameter incremental learning algorithm for neural networks. *neural networks, IEEE Transactions on,* 17**,** 1424-1438.

SHIN, K. & MIYAZAKI, S. 2016. A fast and accurate feature selection algorithm based on binary consistency measure. *Computational Intelligence,* 32**,** 646-667.

SILVESTRE, M. R. & LEE LUAN, L. 2006. Statistical evaluation of pruning methods applied in hidden neurons of the MLP neural network. *Latin America Transactions, IEEE (Revista IEEE America Latina),* 4**,** 249-256.

SOMOL, P., PUDIL, P., NOVOVIČOV , J. & PACLıK, P. 1999. Adaptive floating search methods in feature selection. *Pattern Recognition Letters,* 20**,** 1157-1163.

SPIRKOVSKA, L. & REID, M. B. Connectivity strategies for higher-order neural networks applied to pattern recognition. Neural Networks, 1990., 1990 IJCNN International Joint Conference on, 17-21 June 1990 1990. 21-26 vol.1.

SPIRKOVSKA, L. & REID, M. B. 1993. Coarse-coded higher-order neural networks for PSRI object recognition. *Neural Networks, IEEE Transactions on,* 4**,** 276-283.

SPOREA, I. & GRUNING, A. 2013. Supervised learning in multilayer spiking neural networks. *Neural Computation,* 25**,** 473-509.

SRAVANI, A., HARINI, D. & BHASKARI, D. L. A Comparative study of the classification algorithms based on feature selection. ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol II, 2014. Springer, 97-104.

STRACUZZI, D. J. & UTGOFF, P. E. 2004. Randomized variable elimination. *J. Mach. Learn. Res.,* 5**,** 1331-1362.

SUN, Y., BABBS, C. F. & DELP, E. J. A comparison of feature selection methods for the detection of breast cancers in mammograms: adaptive sequential floating search vs. genetic algorithm. 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, 17-18 Jan. 2006 2005. 6532-6535.

TAKTAK, A. F. G. & LISBOA, P. J. 2006. The use of artificial neural networks in decision support in cancer: A systematic review. *Neural Networks,* 19**,** 408-415.

TAN, C. J., LIM, C. P. & CHEAH, Y. N. 2014. A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models. *Neurocomputing,* 125**,** 217-228.

TING, J. A., D'SOUZA, A., VIJAYAKUMAR, S. & SCHAAL, S. 2010. Efficient learning and feature selection in high-dimensional regression. *Neural Computation,* 22**,** 831-886.

TSAI, C.-F., EBERLE, W. & CHU, C.-Y. 2013. Genetic algorithms in feature and instance selection. *Knowledge-Based Systems,* 39**,** 240-247.

TU, C.-J., CHUANG, L.-Y., CHANG, J.-Y. & YANG, C.-H. 2007. Feature selection using PSO-SVM. *International Journal of Computer Science*.

VUKOVIC, S., DELIBASIC, B., UZELAC, A. & SUKNOVIC, M. 2012. A case-based reasoning model that uses preference theory functions for credit scoring. *Expert Systems with Applications,* 39**,** 8389-8395.

WANG, G., MA, J., HUANG, L. & XU, K. 2012. Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems,* 26**,** 61-68.

WANG, J. Y. & ZHOU, J. 2009. Research of reduct features in the variable precision rough set model. *Neurocomputing,* 72**,** 2643-2648.

WANG, R. & TANG, K. 2009. Feature selection for maximizing the area under the ROC curve. *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops.* IEEE Computer Society.

WANG, R. & TANG, K. 2012. Feature selection for MAUC-oriented classification systems. *Neurocomputing,* 89**,** 39-54.

WASIKOWSKI, M. & CHEN, X.-W. 2010. Combating the small sample class imbalance problem using feature selection. *Knowledge and Data Engineering, IEEE Transactions on,* 22**,** 1388-1400.

WEISS, G. M. & PROVOST, F. J. 2003. Learning when training data are costly: The effect of class distribution on tree induction. *J. Artif. Intell. Res.(JAIR),* 19**,** 315-354.

WERBOS, P. J. 1975. *Beyond regression: new tools for prediction and analysis in the behavioral sciences.* PhD, Harvard University.

WEST, D. 2000. Neural network credit scoring models. *Computers & Operations Research,* 27**,** 1131-1152.

XIAO, J., XIE, L., HE, C. & JIANG, X. 2012. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications,* 39**,** 3668-3675.

YAMADA, M., JITKRITTUM, W., SIGAL, L., XING, E. P. & SUGIYAMA, M. 2014. High-dimensional feature selection by feature-wise kernelized lasso. *Neural Computation,* 26**,** 185-207.

YU, L., YAO, X., WANG, S. & LAI, K. K. 2011. Credit risk evaluation using a weighted least squares SVM classifier with design of experiment for parameter selection. *Expert Systems with Applications,* 38**,** 15392-15399.

YUCHUN, L. 1991. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation,* 3**,** 440-449.

ZENGLIN, X., KING, I., LYU, M. R. T. & RONG, J. 2010. Discriminative semi-supervised feature selection via manifold regularization. *Neural Networks, IEEE Transactions on,* 21**,** 1033-1047.

ZEXUAN, Z., YEW-SOON, O. & DASH, M. 2007. Wrapper-filter feature selection algorithm using a memetic framework. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 37**,** 70-76.

ZHANG, M. 2008. *Artificial higher order neural networks for economics and business,* IGI Global. pp. 1-23.

ZHAO, Z. & LIU, H. 2009. Searching for interacting features in subset selection. *Intelligent Data Analysis,* 13**,** 207-228.

ZHENG, X., JULSTROM, B. A. & CHENG, W. Design of vector quantization codebooks using a genetic algorithm. Evolutionary Computation, 1997., IEEE International Conference on, 1997. IEEE, 525-529.

ZHU, Z., ONG, Y.-S. & DASH, M. 2007. Wrapper–filter feature selection algorithm using a memetic framework. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 37**,** 70-76.